Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Системы коллективного пользования

1990

12

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

12/1990

Издается
ежемесячно
с 1988 г.

СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

В номере:

В. Ф. КОРНЮШКО,
А. Ю. ЖЕДЬ
Системы коллективного пользования на основе персональных компьютеров

РУБРИКИ

Языки программирования
Нам пишут
БК за рога
Как "убить" машинное время

Издательство
"Знание"
Москва
1990

ББК 32.97
С 41

Авторы выпуска

ЧАСТИКОВ Аркадий Петрович — кандидат технических наук, доцент. Научные интересы — эволюция СУБД и экспертных систем.

МАЛЫХИНА Мария Петровна — кандидат технических наук, доцент, программист.

ВОРОНИН Евгений Владимирович — инженер НИС ОМГУ.

КОРНЮШКО Валерий Федорович — кандидат технических наук, доцент, заведует кафедрой вычислительной математики и ЭВМ в МИТХТ им.М.В.Ломоносова. Специалист в области компьютерной техники.

ЖЕДЬ Алексей Юрьевич — ведущий инженер кафедры вычислительной техники МИТХТ им.М.В.Ломоносова.

АВСЕЕВ Александр Владимирович — учащийся московской средней школы № 236. Победитель городского конкурса программ среди школьников, дипломант III всесоюзной олимпиады по информатике 1990 г.

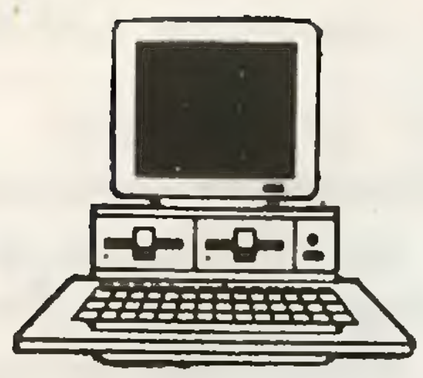
АВСЕЕВ Владимир Владимирович — инженер, специалист в области радиоэлектроники и вычислительной техники.

УСЕНКОВ Дмитрий Юрьевич — студент МГТУ им. Н.Э.Баумана, программист.

РЕДАКТОР Б. М. ВАСИЛЬЕВ

ISBN 5-07-001851-5

© Издательство "Знание", 1990 г.



Во многих организациях уже имеется кое-какая вычислительная техника (в основном ЭВМ серий "Электроника", ДВК, СМ-1800, ЕС1840 и т.п.), которую необходимо объединить в комплекс для решения научных, учебных или информационно-организационных задач. При этом непрофессиональному пользователю придется рассматривать весьма трудные вопросы выбора типа системы, наиболее подходящей для решения его конкретных задач. При этом часто возникают ситуации, когда использование сети ПК может оказаться экономически менее выгодны по сравнению с использованием многотерминальной системы виртуального доступа, рассмотренной в этой статье.

В.Ф.Корнюшко,
А.Ю.Жедь

СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ НА ОСНОВЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

Терминалы

Терминал, как периферийное устройство ЭВМ, представляет собой как бы два устройства: устройство отображения информации (вывода) и устройство ввода информации. Устройство отображения информации чаще всего реализуется на электронно-лучевой трубке, хотя в некоторых случаях им может быть и принтер. Устройством ввода иформации, как правило, бывает алфавитно-цифровая клавиатура. Оба эти устройства соединяются с ЭВМ по каналам связи. В настоящее время наиболее часто для этих целей применяются каналы последовательной передачи информации по стандарту RS-232 или "токовая петля". Как правило, каналы передачи и приема информации независимы друг от друга и могут работать одновременно (дуплексный режим).

Конструктивно дуплексный канал связи представляет собой две витые пары проводов, один конец которых присоединен на вход/выход терминала, а второй — к интерфейсному модулю ЭВМ. Практически число проводов в канале связи обычно больше четырех, поскольку для уменьшения помех и увеличения надежности в кабель вплетаются дополнительные жилы "логической земли", дублирующие "сигнальные" жилы и т.д. Существуют и другие варианты подключения терминалов к ЭВМ с использованием других стандартов и других физических сред передачи сигналов, например, коаксиального кабеля. Коаксиальные и оптоволоконные кабели очень дороги и применяются сейчас в основном в мощных вычислительных сетях. По этим линиям передаются битовые последовательности, которые представляют собой коды символов. Кодирование может производиться в соответствии с различными стандартами. Наиболее часто используется стандарт ASCII (American Standart Code for Informational Interchange) [8].

Следует отметить, что стандартизированными являются только коды от 0(8) до 177(8) ((8) после числа обозначает запись в восьмеричной системе счисления). Коды от 200(8) до 377(8) оставлены для псевдографических символов расширений таблицы ASCII, указывающих национальный признак, и требуют 8-битного представления символов. Обычно же для стандартного набора достаточно 7-битного представления и дополнительная таблица символов кириллицы с кодами 101(8)-176(8). Переключение с таблицы на таблицу осуществляется передачей терминалу от ЭВМ управляющих символов "SO" и "SI" (коды 16(8) и 17(8)).

Набор кодов, которые терминал способен передать в ЭВМ не ограничен теми символами, которые выгравированы на клавишах. С помощью одновременного нажатия на управляющие клавиши (SHIFT, CTRL) можно изменять регистр передаваемого символа, или получать коды в диапазоне 0(8)-37(8) не выводимые на печать. Другие спецклавиши могут генерировать ESC-последовательности,

т.е. последовательности символов, начинающиеся с управляющего символа ESC (код 33(8)) и используемые для управления в таких программах, как текстовые редакторы, меню и т.д.

Попробуем провести классификацию терминалов относительно того, что они отображают и передают. С этой точки зрения терминалы можно разделить на: алфавитно-цифровые, графические, алфавитно-графические и специализированные.

Сейчас наиболее распространены алфавитно-цифровые терминалы. К ним относятся устройства, предназначенные для ввода и отображения только алфавитно-цифровой информации. Большинство используемых в СССР терминалов относится к этой категории. В нее входят как простейшие устройства с бумажной лентой и механизмом пишущей машинки типа Consul, так и достаточно сложные и удобные дисплеи типа ВТА-2000, СМ 7209 и т.д.

Чисто графические терминалы — это большая редкость. В большинстве случаев применяются алфавитно-графические терминалы, предназначенные для работы как с алфавитно-цифровой, так и с графической информацией. Все терминалы такого типа изготавливаются на основе электронно-лучевых трубок с большой разрешающей способностью.

С вводом и выводом алфавитно-цифровой информации все достаточно просто — приходит код символа от ЭВМ — на дисплее отображается символ, нажата клавиша с изображением символа — код символа передается в ЭВМ. Несколько сложнее обстоит дело с передачей графической информации. Дисплей "понимает" определенные последовательности символов как команды на построение векторов, окружностей и т.д. А чтобы при передаче обычной алфавитно-цифровой информации дисплей не воспринял ее как графическую команду, имеется специальная последовательность управляющих кодов, указывающая, что далее будут передаваться графические команды.

Еще сложнее организация ввода графической информации. Только немногие, самые сложные алфавитно-графические терминалы имеют так называемое "световое перо", с помощью которого можно передать в ЭВМ координаты отмеченной им точки экрана. Если же терминал не имеет светового пера, то используется режим ввода алфавитно-цифровой информации с последующей интерпретацией ее программой на ЭВМ по тем или иным правилам.

Конечно, описанный выше способ комбинации (приема/передачи) алфавитно-цифровой и графической информации не единственно возможный, но он наиболее часто используется в реальных системах. Имеется даже специальный набор команд для управления дисплейной графикой, называемой REGIS-графикой.

Теперь рассмотрим специализированные терминалы. С ними каждый из нас сталкивается каждый раз, когда приходит в сберегательную или железнодорожную кассу. Дисплей, стоящий перед кассиром, не имеет никаких особенностей, но если присмотреться к клавиатуре, то можно заметить на ней клавиши, назначение которых нам абсолютно неизвестно, однако кассир их нажимает сравнительно часто. Например, для того чтобы набрать число 5000, кассиру не приходится нажимать на клавиши четыре раза (один раз на 5 и три раза на 0). Он нажмет только на клавишу 5 и на клавишу 1000. Клавиши 100, 1000, 10000 и E+, E- и другие подобные значительно облегчают работу кассиру, но абсолютно не нужны композитору или наборщику. Однако на терминале у наборщика вы найдете другие клавиши, которые необходимы только в его работе. Вот этим в основном и отличаются специализированные терминалы от терминалов общего назначения.

Следует, однако, понимать, что речь идет не о том или ином наборе клавишей, а о том, что специализированным называется терминал, наилучшим образом приспособленный

для какого-либо вида профессиональной деятельности и, как правило, не пригодный для других применений. Кстати, чисто графический терминал тоже может быть отнесен к этой категории.

Теперь попробуем рассмотреть терминалы с точки зрения предоставляемых ими возможностей. С этой позиции терминалы можно разделить на обычные и "интеллектуальные".

Обычных терминалов в настоящее время практически не осталось. Если же терминал способен принимать от ЭВМ и исполнять определенные команды, не связанные непосредственно с вводом/выводом символов, например, смену страниц дисплейной видеопамяти, блокировку клавиатуры и т.д. или выполнять какие-либо действия самостоятельно (буфферизацию вводимых/выводимых символов, выбор скорости приема/передачи, гашение дисплея для предотвращения выгорания монитора при длительном отсутствии ввода/вывода), то такой терминал называют интеллектуальным. Практически все современные терминалы обладают той или иной степенью интеллектуальности. Некоторые по существу представляют собой микро-ЭВМ со своим микропроцессором (часто с несколькими микропроцессорами), ОЗУ, ПЗУ, CMOS-памятью и другими аксессуарами. Однако надо всегда помнить, что большие возможности всегда сопровождаются толстенными томами инструкций для пользователя, поэтому при выборе типа терминала нужно придерживаться принципа разумной достаточности.

Однозадачные многотерминальные системы

Многотерминальная система включает в себя ЭВМ и несколько терминалов, подключенных к этой ЭВМ с помощью интерфейсных устройств. Топология подключения терминалов может быть различной: звезда, кольцо, шина (рис.1). Это зависит от типа используемой ЭВМ и интерфейсных устройств. Однако во всех случаях в этих системах существует один общий признак, позволяющий отнести данную систему к разряду многотерминальных систем: все без исключения процессы обработки информации выполняются только на центральной ЭВМ, а процессы, происходящие на линиях связи и в терминалах, связаны только с ее передачей, приемом и отображением. Другими словами, терминалы с их линиями связи служат только для ввода и вывода ин-

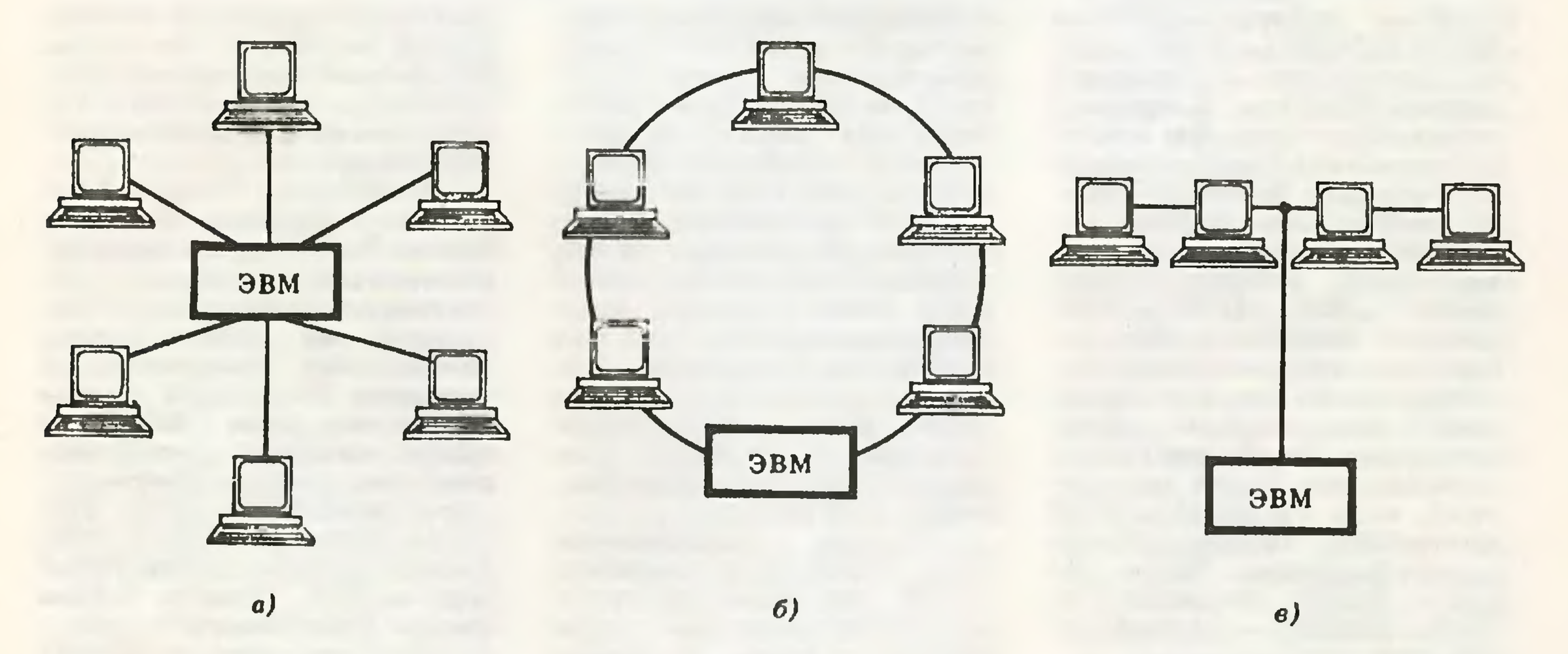


Рис.1. Топология систем коллективного пользования:
а) звездообразная; б) кольцевая; в) шинная

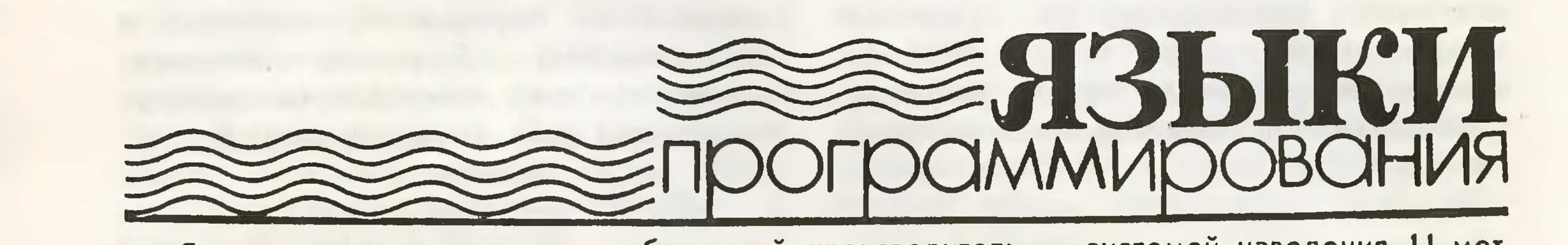
формации, а обработка ее ведется на ЭВМ. Если это условие не выполняется, то речь уже идет не о многотерминальной системе, а о системе с распределенной обработкой информации, системе виртуального доступа или о сети ЭВМ.

Как же фунционирует многотерминальная система? На ЭВМ система запускается с помощью специальной программы-диспетчера. В задачу этой программы входит реализация так называемой дисциплины обслуживания терминалов, т.е. обеспечения получения информации с терминалов и вывода информации на них. Здесь возможно огромное число вариантов, которым соответствуют определенные термины, характеризующие систему. Рассмотрим некоторые из них.

В качестве примера возьмем вариант системы, обеспечивающей контроль знаний учащихся в ВУЗе. Допустим, к центральной ЭВМ каналами связи присоединены десять терминалов, расположенных на рабочих местах учащихся. Еще один терминал установлен на рабочем месте преподавателя. Двенадцатый терминал (сервисный) установлен непосредственно рядом с ЭВМ и используется обслуживающим персоналом для контроля функционирования всей системы, ее запуска и останова. Такая конфигурация технических средств обычно поставляется в составе вычислительных комплексов на базе широко распространенных ЭВМ СМ1300.01, СМ4, СМ1420. Максимально возможное количество терминалов зависит от типа ЭВМ, объема ее ОЗУ и типа используемых интерфейсных устройств и должно соответствовать классу решаемых задач. Так, к ЭВМ СМ1420 с объемом ОЗУ 2 Мб может быть присоединено до 32 терминалов СМ7209 или 7222.01.

Подключение большего числа терминалов без увеличения объема ОЗУ и быстродействия процессора нецелесообразно, так как приводит к резкому снижению скорости реакции системы.

На ЭВМ функционирует программа-диспетчер, в задачу которой входит выдача вопросов на терминалы учащихся, получения ответов на эти



Язык программирования ФОРТ (FORTH — в переводе с английского — "вперед") создан Чарльзом Х.Муром в конце 60-х годов. Вот что он рассказывает относительно необычности названия своего языка: "Свои первые результаты я получил в то время, когда работал с моделью IBM 1130 — ЭВМ третьего поколения. Они показались мне настолько значительными, что я посчитал новый язык языком машин четвертого (fourth) поколения и назвал его Fourth (четвертым), если бы модель 1130 допускала пятисимвольные идентификаторы. Таким образом Fourth превратился в Forth (ФОРТ) — своеобразная игра слов".

По словам Ч.Мура, основной целью создания нового языка было повышение собственной производительности. Он пишет: "Традиционные языки программирования не обеспечивали желаемой для меня мощности, легкости и гибкости. Я подсчитал, что за 40 лет очень хороший программист может написать 40 программ, а мне хотелось гораздо больше. В мире столько вещей, которые нужно сделать, что мне требовался инструмент, способный помочь в этом" (по словам Мура, в своем окончательном виде новый язык повысил его производительность в 10 раз).

Первым применением языка ФОРТ стала созданная в 1971 году Муром программа для Национальной радиоастрономической обсерватории в г.Китт-Пик, шт.Аризона, которая в реальном масштабе времени управляла системой наведения 11-метрового телескопа. Затем им был разработан первый компилятор, заменивший используемый для этого кросс-ассемблер.

В 1973 году Ч.Мур, Э.Разер и Н.Конклин основали фирму Forth Inc., которая тиражировала системы, пригодные не только для обсерваторий, но и для других специальных применений. В середине 70-х годов фирма поставляла язык ФОРТ на гибких магнитных дисках для реализации на микропроцессорах Intel 8080, 8086, Z80, LSI-11, Motorola 6800, TI 9900, а также для мини-ЭВМ PDP-II фирмы DEC, Nova и Eclipse фирмы Data General и др.

Что же такое ФОРТ? "ФОРТ — необычный язык. Он "попирает" многие устоявшиеся правила программи-

вопросы, анализ ответов и вывод результатов на терминал преподавателя и, возможно, на печатающее устройство. Кроме того, программа должна быть готова к приему корректирующих воздействий с терминала преподавателя и с сервисного терминала. В данной ситуации такое построение системы и программы-диспетчера абсолютно логично, поскольку обеспечивает полный контроль над терминалами учащихся и дает свободу действий преподавателю и обслуживающему персоналу. Терминалы учащихся в этом случае работают в так называемом подчиненном режиме, т.е. ввод с них какой-либо информации возможен только в те моменты, когда программа-диспетчер разрешит этот вывод. С терминалов же преподавателя и с сервисного терминала возможен незапланированный ввод, который ожидается программой-диспетчером в любой момент времени и на который она должна немедленно отреагировать. Такая дисциплина обслуживания позволяет программе-диспетчеру выбирать неповторяющиеся вопросы из базы данных и вести базу данных ответов и результатов проверки.

Таким образом мы видим, что вся система фукционирует под управлением одной программы (задачи), обеспечивая работу многих пользователей за терминалами. Описанная программа может быть реализована на ЭВМ указанных выше типов в среде операционной системы RT-11 (Рафос, Фодос).

Несколько слов необходимо сказать о том, как, собственно, устроена программа-диспетчер. Каким образом удается добиться иллюзии, что все терминалы работают одновременно, печатая и принимая символы? На самом деле ввод и вывод на разные терминалы происходит не одновременно, но очень быстро, с небольшой разницей во времени. Программа-диспетчер при выводе на терминалы отправляет очередной символ на линию, связанную с одним из терминалов, и не ждет, пока этот символ дойдет до терминала, будет отображен и линия освободится. Она просто "забывает" обо всем этом и ищет очередной терминал, на кото-

ФОРТ

рования" — начинает свою книгу по ФОРТу Л.Броуди. Основным понятием ФОРТа является слово, под которым понимается любая последовательность символов (включая и знаки препинания), отличных от пробелов. Язык имеет простой синтаксис. Программа представляет собой последовательность отдельных слов, воспринимаемых как текст печатного издания. Семантика отдельного слова в программе на момент ее исполнения определена либо реализацией (множеством обязательных слов согласно существующему стандарту), либо программистом, который может определять новые слова, пользуясь уже известными системе. Это определение, помеченное специальным символом, записывается как обычная последовательность в тексте. ФОРТ допускает использование кириллицы, символьную обработку, рекурсию, работу с любыми внешними устройствами и обладает возможностями ряда других языков.

Из характерных особенностей языка ФОРТ выделяют следующие:

- наличие стека данных, применяемого для обмена аргументами и отдельными словами-функциями;
- возможность работы с объектами типа "словарь", описывающими активный в данный момент список слов, известный системе;
- динамическое (в момент использования) переключение режимов "интерпретация-компиляция", которое позволяет реализовать сложные структуры памяти и управления;
- наличие двух уровней расширения языка с помощью так называемых определяющих слов и с помощью специального слова ":" (двоеточие);
- встроенный в ФОРТ язык Ассемблер позволяет эффективно реализовывать аппаратно зависимые функции и функции сопряжения с другими системами программирования. К недостаткам языка относят: необычность выражений и отсутствие арифметики с плавающей точкой; отсутствие типов; трудность чтения текстов программ; трудности перевода программ на другие языки; использование польской записи и необычной формы предложения IF... ELSE... THEN.

Появление ФОРТа, как отмечают некоторые ученые, в частности А.Вулф, было воспринято профессионалами неоднозначно. Одни высказывали сомнение, а язык ли это вообще и если язык, то низкого или высокого

рый нужно послать символ и у которого свободна линия. Найдя такой терминал, она посылает ему сигнал-реходит к следующему и т.д., пока не вернется к первому терминалу, который уже "съел" свой символ и освободил линию. Прием символов с клавиатуры осуществляется аналогичным образом. Все это оказывается возможным благодаря тому, что быстродействие процессора ЭВМ во много раз превышает быстродействие линий связи, скорость отображения символов на дисплее и скорость ввода символов человеком с клавиатуры.

Этот же принцип относится и к любым другим периферийным устройствам, будь то устройство печати, магнитные диски или ленты.

Многозадачные многопользовательские системы

Рассмотрим ту же конфигурацию технических средств, что и в предыдущем примере: центральная ЭВМ и 12 терминалов. За терминалами работают программисты: кто-то редактирует текст программы, кто-то транслирует свою программу, кто-то запускает на выполнение уже отлаженную программу и т.д. Если наблюдать за этой работой со стороны, то создается впечатление, что всех их ЭВМ обслуживает одновременно, как и в предыдущем примере. Такой эффект достигается за счет умелого распределения ресурсов управляющей программой-диспетчером. Попробуем в упрощенном виде представить механизм работы этой программы.

Программа-диспетчер находится в постоянном внимании к каждому терминалу, например, периодически проверяя был ли произведен какой либо ввод с терминала. Как только программист, работающий за терминалом набирает на клавиатуре команду (вызов редактора, транслятора, своей программы), эта команда немедленно поступает к программе-диспетчеру, которая выполняет загрузку вызванной программы в ОЗУ ЭВМ и запускает ее. С этого момента программист за терминалом общает-

---

уровня; вторые считали ФОРТ вехой в развитии языков программирования; третьи — шагом в сторону. Достойное место среди языков программирования он занял после ряда успешных реализаций на мини- и микро-ЭВМ. С появлением персональных компьютеров кривая популярности ФОРТа резко подскочила вверх, и по некоторым утверждениям, вывела его к середине 80-х годов на третье место после БЕЙСИКа и ПАСКАЛЯ.

В 1977 году Ким Харрис из Кремниевой долины организовал группу под названием ФИГ (FORTH Interest Group), которая в следующем году создала упрощенный интерпретатор языка для персональных компьютеров под названием ФИГ-ФОРТ (FIG-FORTH), продававшийся по очень низкой цене. ФИГ-ФОРТ завоевал огромную популярность у массового пользователя. У нас в стране система ФИГ-ФОРТ реализована на мини-ЭВМ СМ-4 с операционными системами ОС РВ и РАФОС.

Язык ФОРТ нашел применение во многих областях науки и техники, в частности при разработке трансляторов и операционных систем; для сбора и анализа данных; в системах машинной графики; при создании экспертных систем; для управления станками и роботами и т.д.

Хорошая реклама ФОРТу была сделана летом 1985 года, когда после 16 дней упорных поисков франко-американская научная экспедиция с помощью подводного аппарата "Арго" в водах Северной Атлантики на глубине более 3 км обнаружила затонувший в 1912 году океанский лайнер "Титаник". Компьютерная система, установленная на подводном аппарате и предназначенная для автоматического управления видеокамерами, работала по программе, написанной на языке ФОРТ.

В настоящее время как за рубежом, так и в нашей стране осуществлены и успешно развиваются аппаратные реализации языка ФОРТ, так называемые ФОРТ-процессоры (один из первых выпустил Ч.Мур), которые выполняют слова ФОРТа как машинные команды.

В 1983 году был выпущен стандарт на язык ФОРТ — "ФОРТ-83".

В США с 1978 года издается периодический журнал "FORTH Dimensions", предназначенный для массового пользователя. А для программистов-профессионалов с 1983 года выходит журнал "The Journal of Forth Application and Research". Начиная с конца 70-х годов проводятся ежегодные конференции, посвященные проблемам развития ФОРТ-систем.

В нашей стране первая книга по языку ФОРТ вышла в 1988 году [2], хотя работы по реализации языка на отечественных компьютерах начались раньше. Среди реализованных систем необходимо отметить следующие.

Единый ФОРТ для микропроцессоров К580, К1801, К1810 разработан сотрудниками ЛГУ и НИИСчетмаша. Система установлена на СМ-

ся не с программой-диспетчером, а со своей вызванной программой. Кроме этих действий программа-диспетчер при запуске программы пользователя помещает запись о ней в начало своей внутренней очереди. Через определенный интервал времени (обычно около 0.1 с) выполнение программы пользователя прерывается,управление передается программе-диспетчеру, которая перемещает запись о запущенной программе из начала очереди в конец, выбирает из начала очереди информацию о другой задаче (запущенной ранее другим программистом) и запускает ее. Или же выполняет запрос на запуск другой программы, который мог появиться, если к этому моменту какой-либо программист закончил ввод команды. Этот процесс снова повторяется через тот же интервал времени. Таким образом, время центрального процессора ЭВМ равномерно разделяется между всеми задачами системы.

На самом деле описанный процесс является только частью того алгоритма, по которому работает программа-диспетчер. Например, если очередная задача, находящаяся в голове очереди, в этот момент ожидает ввода каких-либо данных с терминала, то ее не обязательно запускать немедленно и отдавать ей квант процессорного времени. Она все равно будет находиться в режиме ожидания до тех пор, пока ввод не будет завершен. Запись о такой задаче переставляется в хвост очереди, а на ее место ставится следующая задача из головы очереди. Кроме того, могут возникать ситуации, связанные с ограниченностью объема ОЗУ. Если очередная, вызванная с терминала, задача не помещается в ОЗУ, то программа-диспетчер может просмотреть очередь, выбрать задачи, которые ожидают завершения ввода или вывода и выгрузить их из ОЗУ на диск, а вместо них запустить требуемую задачу. Как только ввод/вывод для задачи, выгруженой на диск, будет завершен, на диск будет выгружена другая задача, ожидающая завершения ввода/вывода (если к этому моменту нет достаточного свободного пространства ОЗУ), а выгру-

---

3, СМ-4, СМ-1800, ДВК, "Электроника-60", ЕС-1840, "Корвет" и др. Система ФОРТ-ЕС разработана С.Н.Барановым в ЛИИАНе для ЕС ЭВМ и работает под управлением операционных систем ОС ЕС и СВМ ЕС. Система Астро-ФОРТ разработана сотрудниками ИТА АН СССР и ЛИИАН для ПЭВМ ЕС 1840. Система ФОРТ "Искра-226" создана в Институте социально-экономических проблем АН СССР. Система ФОРТ-М6000 разработана ленинградцем В.Н.Патрышевым для ЭВМ М6000, она ориентирована на стандарт "ФОРТ-83". Система ФОРТ-БЭСМ-6, созданная И.Р.Агамирзяном (ИТА АН СССР и ЛИИАН) для ЭВМ БЭСМ-6, работает под управлением операционной системы ДИСПАК. Система ФОРТ-Эльбрус, разработанная в ЛГУ А.Е.Соловьевым для МВК "Эльбрус", работающая под управлением ОС "Эльбрус". Система ФОРТ-Тарту, созданная сотрудниками ВЦ Тартуского государственного университета, работает под упраением ОС РВ и UNIX.

При Комиссии по технологии программирования при ГКНТ в рамках рабочей группы по технологии программирования в 1985 году была создана целевая подгруппа по языку ФОРТ, в задачи которой входило распространение опыта использования ФОРТ-систем.

В последние годы ленинградские разработчики ФОРТ-систем создали научно-производственный кооператив "ФОРТ-Инфо" для развития и распространения ФОРТ-систем в различных областях человеческой деятельности.

Приведем в заключение еще одно высказывание автора языка: "ФОРТ представляет человеку удобные средства общения с "умными машинами", которыми он себя окружает. Следовательно, этот язык должен обладать свойствами естественных языков, включая компактность, разносторонность и расширяемость. Я считаю его лучшим языком для написания программ, изложения алгоритмов или объяснения работы компьютера...".

Литература

1. *Хелмс Г.Л. Языки программирования/Пер.с англ. — М.: Радио и связь, 1985. — 176 с.*
2. *Баранов С.Н., Ноздрунов Н.Р. Язык ФОРТ и его реализация. — Л.: Машиностроение, 1988. — 157 с.*
3. *Порошин А.Н., Тозик В.Т., Юдин Ю.В. Архитектура программного обеспечения САПР изделий машиностроения. — Л.: ЛДНТП, 1988.*
4. *Бураго А.Ю., Кириллин В.А., Романовский Н.В. ФОРТ-язык для микропроцессоров. — Л.: Знание, 1989. — 36 с.*
5. *Язык компьютера/Пер. с англ. Под ред. В.М.Курочкина. — М.: Мир, 1989. — 240.*
6. *Броуди Л. Начальный курс программирования на языке ФОРТ/Пер. с англ. — М.: Финансы и статистика, 1990. — 352 с.*

М.П.Малыхина
А.П.Частиков

женная ранее задача будет загружена и запущена.

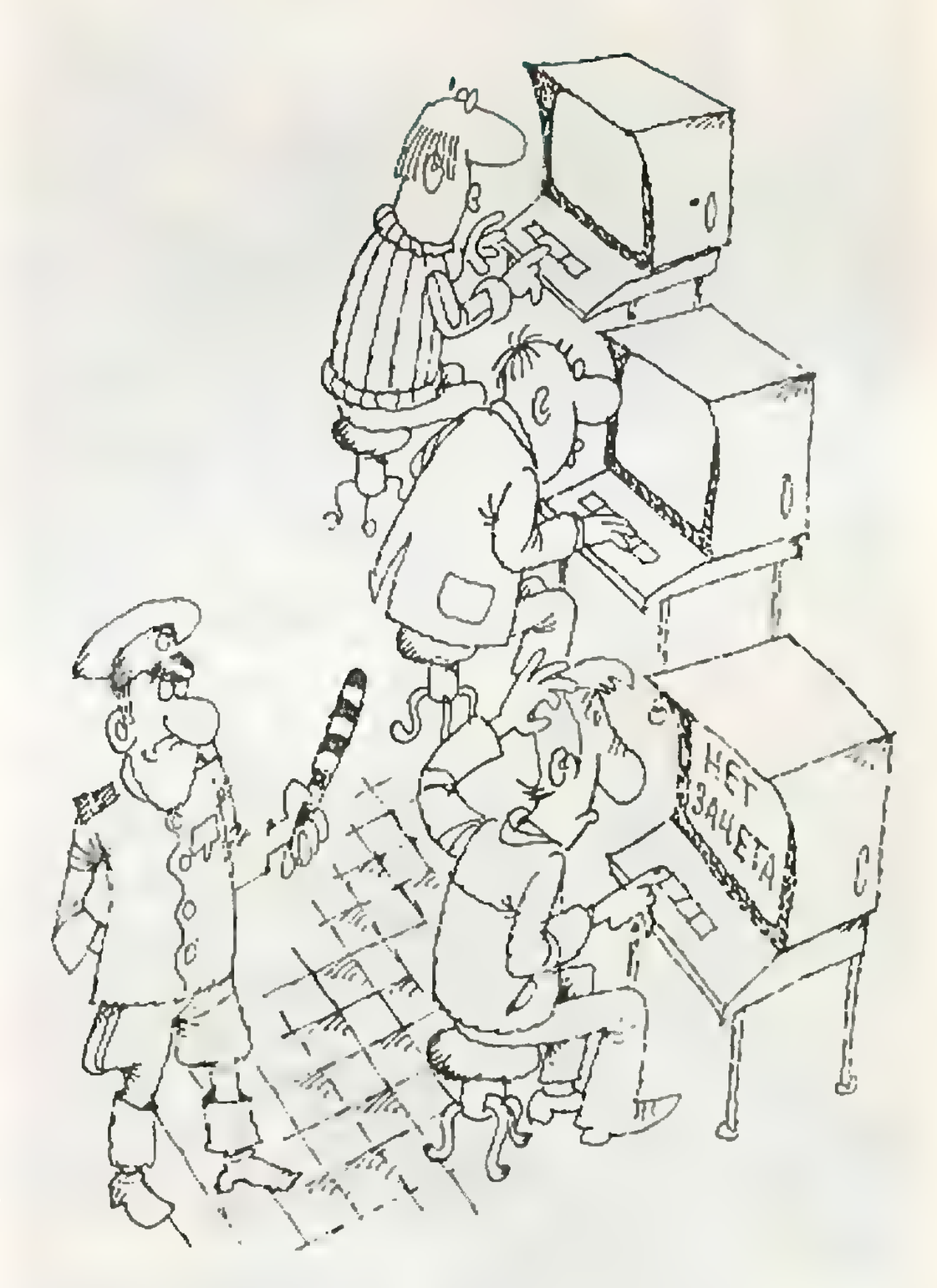
Кроме этого, программа-диспетчер может учитывать приоритеты задачи. Существуют задачи, предназначенные для немедленного запуска при какой-либо непредвиденной ситуации, например, задача уплотнения памяти при чрезмерной ее сегментации. Программа-диспетчер следит за такими ситуациями и осуществляет запуск необходимых задач.

Как видим, вариант многопользовательской многозадачной системы много сложнее варианта многопользовательской однозадачной. Однако возможности такого варианта значительно богаче.

Для ЭВМ типа СМ-1300.01, СМ-4, СМ-1420 существует операционная система ОС-РВ, в которой реализованы механизмы многопользовательской многозадачной работы. Кроме того, для ЭВМ этих типов имеются варианты ОС UNIX, распространяемые под названиями ИНМОС и ДЕМОС. В последнее время в литературе (см.[8]) появились сообщения о создании многотерминальных систем на базе IBM PC AT. При использовании в ЭВМ процессора 80286 к ней можно подключать до десяти терминалов, а в случае использования процессора 80386 — до тридцати двух. Эта возможность появилась в результате разработки новой операционной системы PICK.

Существуют и другие варианты организации работы многопользовательских систем, но все они в той или иной степени похожи на два основных варианта, описанных выше.

Приведем еще примеры многопользовательских систем. Многопользовательская однозадачная система может быть удачно применена для организации системы сдачи экзаменов по правилам дорожного движения в ГАИ. В настоящее время для этих целей применяются достаточно сложные электромеханические или электронные устройства. Перед экзаменующимся на экране устройства высвечивается очередной вопрос и



несколько вариантов ответа. Он должен выбрать правильный ответ и нажать кнопку с соответствуюшим номером. По завершении серии вопросов выставляется отметка (обычно по системе сдал/не сдал), что фиксируется загоранием соответствующего индикатора на панели прибора. Контроль осуществляется сотрудником государственной автоинспекции, который постоянно находится в комнате, где установлена аппаратура, и фиксирует результаты экзаменов. Кроме того каждое устройство может быть связано с центральным пультом, где также отображаются результаты сдачи экзамена на экране центрального компьютера.

При использовании современных компьютерных систем для этой цели возможны три варианта: 1.оборудование каждого рабочего места экзаменующегося персональным компьютером; 2.оборудование каждого рабочего места экзаменующегося персональным компьютером и все они связаны в локальную вычисли-

тельную сеть; 3.оборудование класса многотерминальной системой.

Первый вариант абсолютно неприемлем, так как повторяет все недостатки электромеханического варианта. Второй вариант более приемлем с точки зрения обеспечения контроля, но он несет в себе большую избыточность, т.е. потенциальная мощность такой системы при решении нашей задачи не будет использована даже на половину. С экономической точки зрения этот вариант также не выгоден так как цены сетевого аппаратного и математического обеспечения сопоставимы с ценами самих ПК. Кроме того, обсуживание локальной сети требует высококвалифицированного персонала — программистов и электронщиков.

Остается вариант с многотерминальной системой. Рассмотрим его. В качестве головной ЭВМ достаточно использовать не слишком мощную ЭВМ с объемом ОЗУ до 64 Кб и одним накопителем на жестком диске. Кроме того, ЭВМ должна быть укомплектована устройством печати, консольным терминалом и мультиплексором, который обеспечит связь с 32 терминалами, расположенными в классе на рабочих местах экзаменующихся. В качестве терминалов можно использовать обычные алфавитно-графические терминалы либо специализированные с упрощенным набором клавиш. Для наших целей достаточно иметь клавиатуру с набором из 7-8 клавиш: цифровые клавиши от 1 до 5 — для выбора ответа, клавишу сброса — для инициализации сеанса работы, клавишу запроса помощи. Возможны и другие, но более или менее похожие наборы. Стоимость такой системы вполне сопоставима со стоимостью применяемых в настоящее время систем электромеханических экзаменаторов, а стоимость обслуживания и эксплуатации значительно ниже.

Работа с описываемой системой происходит следующим образом. Экзаменатор запускает ЭВМ и инициализирует систему с консольного терминала, установленного на его рабочем месте. Очередной экзаменующийся подходит к экзаменатору. На дисплее экзаменатора высвечивается запрос данных экзаменуемого. Экзаменатор вводит данные экзаменующегося, и программа проверяет наличие этих сведений в своей базе данных. Ответ программы может быть нескольких видов в зависимости от ситуации. Если экзаменующийся сдает экзамен в первый раз и сведений о нем в базе данных нет, то эти сведения записываютя в базу данных и делается пометка о текущей дате.

Если же экзаменующийся пришел на экзамен повторно, но ранее необходимого срока, то на дисплее перед экзаменатором появится соответствующая информация и человек не будет допущен до экзамена. Возможен еще один вариант, когда экзаменующийся исчерпал допустимый лимит попыток. В этом случае он будет вынужден повторить курс обучения.

Будем считать, что в нашем примере все прошло благополучно и экзаменующийся занял место за терминалом. Экзаменатор начинает вводить данные следующего экзаменующегося, а перед человеком за терминалом на дисплее появляется надпись: "Когда Вы будете готовы, нажмите клавишу "Сброс".Действие выполнено и перед экзаменующимся на дисплее появляется первый вопрос и пять ответов к нему.

Вопрос выбирается программой таким образом, чтобы на близлежащих дисплеях он не дублировался. Экзаменующийся должен выбрать правильный ответ и нажать соответствующую цифровую клавишу.

Правильность или неправильность ответа экзаменующемуся не сообщается. После того как он ввел номер ответа, на экране появляется следующий вопрос и т.д. По завершении серии вопросов на экране появляется надпись "Экзамен сдан/не сдан" в зависимости от результата. Экзаменатор в любой момент времени может проверить состояние рабочего места экзаменующегося, т.е. просмотреть количество правильных и неправильных ответов, получить у себя на экра-

не копию вопроса, на который в данный момент времени отвечает экзаменующийся. По завершении экзамена результаты могут быть распечатаны на устройстве печати.

Очевидно, что при решении этой конкретной задачи можно, как и указывалось выше, использовать как набор ПК, так и сеть ПК. Однако преимуществом многотерминальной системы является ее относительная дешевизна (одна не слишком мощная ЭВМ с ограниченным набором периферийных устройств и дешевые специализированные терминалы), возможность организации контроля за всеми сдающими экзамен, а также возможность быстрого оперативного внесения любых корректив в программу экзаменов с рабочего места преподавателя при простоте и дешевизне математического обеспечения. Отметим и преимущества компьютерной автоматизированной системы сдачи экзаменов перед старой электромеханической системой. Во-первых, простота подготовки экзаменационных вопросов и занесение их на машинные носители. Во-вторых, создание баз данных на экзаменующихся на любой срок и хранение этой информации на внешних ЗУ. В-третьих, ведение протоколов экзаменов на машинных носителях, т.е. организация "безбумажной" технологии ведения документации. И, наконец, в-четвертых, переход в недалеком будущем от примитивных контролирующих систем к контролю с элементами подсказки и обучения, а затем к обучающим и контролирующим системам.

Еще один пример — это справочно-информационная система, строящаяся на основе многозадачных многотерминальных систем. Основное их назначение — отображение текущей информации на дисплеях, расположенных либо в пределах одной организации (аэропорт), либо в пределах городского района.

В пределах городского района или поселка может быть организована справочная система типа "электронной газеты". Подписчики такой газеты снабжаются приставками к телевизору, состоящими из клавиатуры и модуля сопряжения с телефонной линией. Связь абонента с ЭВМ осуществляется по телефонному каналу. Абонент может вводить с клавиатуры запросы на информацию по интересующим его темам (сводка происшествий, время работы магазинов и наличие товаров, последние новости и т.д.) и получать изображение соответствующего текста на экране телевизора.

Список литературы

1. Кейслер С. Проектирование операционных систем для малых ЭВМ. — М.: Мир, 1986. — 680 с.

2. Операционная система ОСРВ СМ ЭВМ. — М.: Финансы и статистика, 1987. — 271 с.

3. Инструментальная мобильная операционная система ИНМОС. — М.: Финансы и статистика, 1985. — 231 с.

4.Дамке М. Операционные системы микро-ЭВМ. — М.: Финансы и статистика, 1985. — 151 с.

5. Мир ПК. — 1990. — № 1. — С.69.

6. Thomas R., Yates J. A User Guide to the UNIX System, Osborne, McGraw Hill, Berkeley, California: — 1985. — 716 p.

7. Jukari Shirota, Tosijasu L. Kunii, First Book on UNIX for Executives, Springer-Verlag. — Tokyo: — 1984. — 154 p.

8. Seyer, Martin D. DOS/UNIX Systems. Prentice Hall: — 1986. — 262 p.

Системы виртуального доступа

Системы виртуального доступа занимают промежуточное положение между многотерминальными системами и сетями ЭВМ. На сегодня четкого определения понятия "системы виртуального доступа" нет не только в популярной, но и в технической литературе, хотя само название и описание существующих реальных систем встречается довольно часто (например [1], [2]).

Обычно системы виртуального доступа состоят из одной центральной ЭВМ, выполняющей функции диспетчера системы, и файл-сервера, т.е.

устройства, обеспечивающего доступ к централизовано расположенным файлам всех абонентов системы, и некоторого количества периферийных ЭВМ. Топология системы виртуального доступа может быть такой же, как и топология многотерминальных систем (звезда, кольцо, шина), но может принимать и весьма замысловатые очертания типа дерева или многосвязной сети (рис 2).

Основная отличительная черта системы виртуального доступа, по сравнению с многотерминальной системой в том, что абонент системы имеет в своем распоряжении полноценную ЭВМ (обычно ПК), и все вычислительные процессы и процессы обработки информации идут на этой периферийной ЭВМ, а не на центральной ЭВМ, как в случае многотерминальных систем. В то же время все данные для работы программ, да и сами программы распологаются на дисках, подключенных к удаленной (центральной) ЭВМ, функции которой ограничены обеспечением доступа к этим данным и программам.

Если же сравнить системы виртуального доступа с сетями ЭВМ, то окажется, что системы виртуального доступа отличаются от сетей ЭВМ тем, что практически не выполняют (или выполняют в очень ограниченном объеме) функций по взаимодействию пользователей друг с другом (например, электронной почты). Однако стоимость аппаратного и программного обеспечения систем виртуального доступа значительно ниже стоимости сетей ЭВМ, что достигается благодаря целого ряда факторов.

Центральная ЭВМ в системах виртуального доступа — это обычно ЭВМ средней производительности, с минимально необходимым объемом ОЗУ, укомплектованную внешними запоминающими устройствами большой емкости и другими периферийными устройствами. В ее функции входит удовлетворение запросов периферийных ЭВМ на чтение/запись на все внешние устройства и диспетчерские функции, т.е. определение прав доступа периферийных ЭВМ к тому или иному устройству, построение приоритетных очередей при наличии нескольких запросов к одному и тому же устройству и т.д. По сути, каждый пользователь системы виртуального доступа имеет в своем распоряжении персональный компьютер и полный набор периферийных устройств (жесткий диск, флоппи-диски,

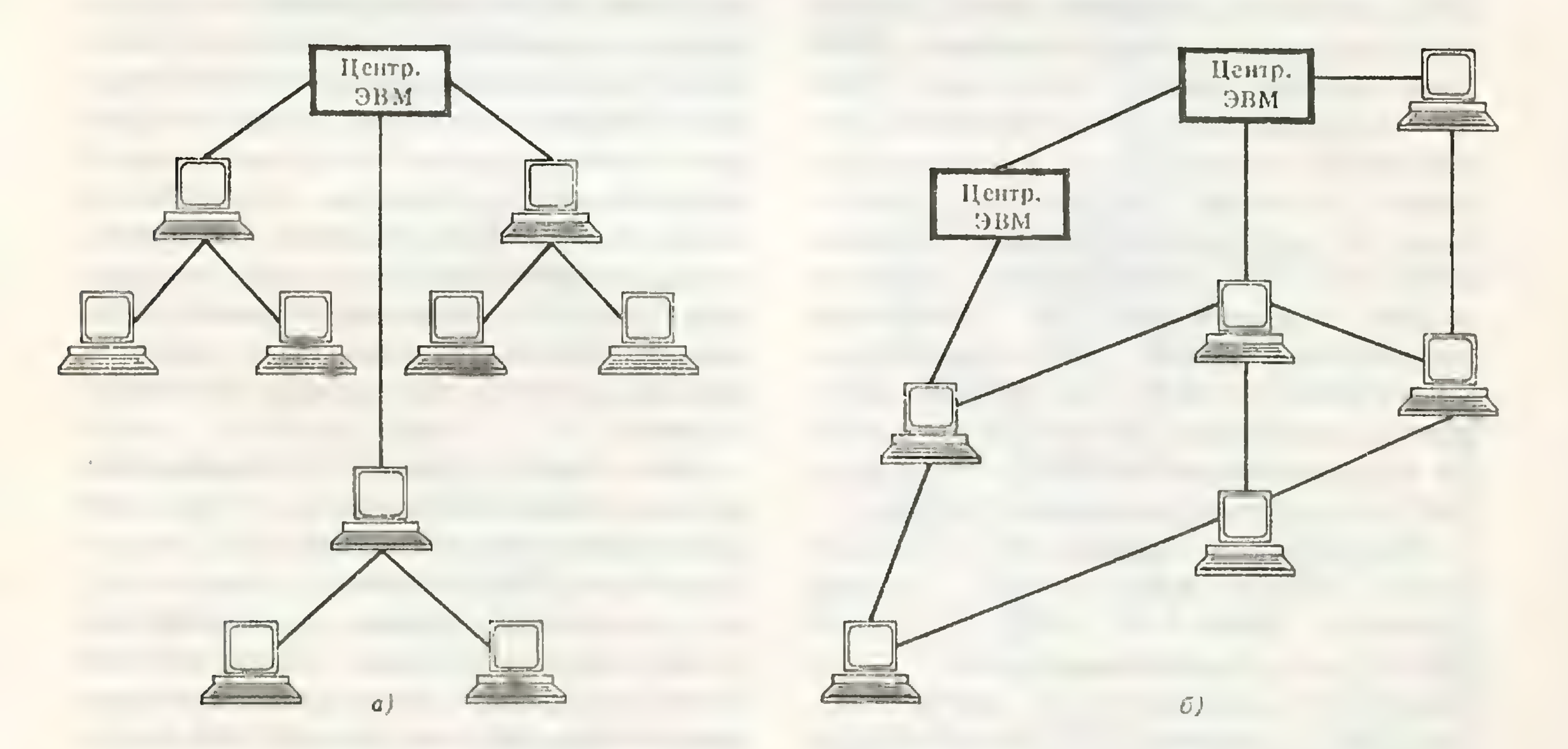


Рис.2. Топология систем коллективного пользования: а) древовидная; б) многосвязная

принтер), хотя на самом деле к его компьютеру подключен только канал связи с центральной ЭВМ. Математическое обеспечение системы виртуального доступа предоставляет пользователям периферийных ЭВМ возможность работать с устройствами головной ЭВМ так же, как если бы эти устройства были подключены к их собственному ПК.

Периферийные ЭВМ системы виртуального доступа могут обладать значительными вычислительными мощностями, и в то же время они не на много превышают по цене обычные интеллектуальные терминалы, поскольку вместо дорогостоящих дисковых запоминающих устройств они оснащены дешевым адаптером связи с центральной ЭВМ. Программы, запускаемые на периферийных ЭВМ, получают свои данные не с дисков, подключенных непосредственно к периферийной ЭВМ (поскольку их просто нет), а по линии связи с дисков головной ЭВМ. Математическое обеспечение системы виртуального доступа организует для программ, запускаемых на периферийных ЭВМ, так называемый "прозрачный" режим доступа к дискам центральной ЭВМ, т.е. программы рассматривают диск (или часть диска) центральной ЭВМ как диск, подключенный непосредственно к периферийной ЭВМ (виртуальный диск). Таким же образом имитируются (эмулируются) печатающие устройства и другая периферия, которой реально не существует на периферийной ЭВМ. Следовательно все программы, которые могут выполняться на полностью укомплектованной периферийными устройствами ЭВМ, могут быть запущены в системе виртуального доступа на периферийных ЭВМ без каких-либо переделок самих программ.

Простейшим примером использования систем виртуального доступа является пакет LAP LINK PLUS для IBM PC — совместимых ЭВМ. Пакет позволяет обращаться к дискам одной ЭВМ с другой ЭВМ по обычному последовательному каналу связи RS-232, причем возможно не только копирование файлов, но и работа с редакторами текстов, базами данных и т.д.

К более сложному примеру системы виртуального доступа относится система СВД-ВУЗ. Система состоит из головной ЭВМ ДВК-4 (МС 0507.04), укомплектованной накопителем на винчестер-диске (20-40 Мб), двумя накопителями на флоппи-дисках по 800 Кб каждый, устройством печати и графопостроителем и шести периферийных персональных ЭВМ УК-НЦ в бездисковой конфигурации.

Связь с центральной ЭВМ осуществляется по витой паре проводов или плоскому кабелю со скоростью 57.6 Кбод. На головной ЭВМ функционирует программа-диспетчер, которая обеспечивает доступ с периферийных ЭВМ к внешним устройствам центральной ЭВМ. С помощью имеющейся в ПЗУ УКНЦ программы-загрузчика по каналам связи осуществляется загрузка операционной системы в периферийные ЭВМ УКНЦ. После этого все периферийные ЭВМ работают совершенно независимо друг от друга. Для каждого пользователя периферийной ЭВМ она представляется персональным компьютером с собственными дисками, устройством печати и т.д. Пользователи системы могут обращаться к базам данных на дисках центральной ЭВМ, запускать готовые или разрабатывать свои программы. При этом вся их деятельность может быть проконтролирована менеджером системы с центральной ЭВМ. Однако общение пользователей системы друг с другом в этой системе затруднено. Возможно только использование разделяемых файлов по методу почтовых ящиков.

Как видим система виртуального доступа позволяет использовать дешевые (лишенные внешних ЗУ) периферийные ЭВМ (сопоставимые по цене с интеллектуальным терминалом), и получать при этом значительные вычислительные мощности, равные практически сумме мощностей периферийных ЭВМ.

Принцип виртуального доступа

может быть использован для связи между ЭВМ и в тех случаях, когда периферийные ЭВМ системы полностью укомплектованы внешними устройствами. Его преимущество заключается в предоставлении "прозрачности" доступа к удаленным файлам, что позволяет использовать существующее прикладное матобеспечение без переделки его для применения в сети. Примером такой системы является система "Алиса", имеющая свободную топологию и реализованную на базе ЭВМ серий СМ-4, СМ-1420, "Электроника 60". В этой системе любая ЭВМ может получить доступ к файлам, расположенным на дисках любой другой ЭВМ системы, при условии, что этот доступ разрешен пользователем этой ЭВМ. Кроме того, в системе "Алиса" реализован также режим виртуального терминала, т.е. иммитируется работа за терминалом удаленной ЭВМ.

В последнее время элементы системы виртуального доступа все чаще включаются в матобеспечение локальных вычислительных сетей.

Где же реально можно наиболее эффективно использовать системы виртуального доступа?

Рассмотрим вариант организации участка наладки и контроля одноплатных ЭВМ. На плате такой ЭВМ установлены микропроцессор (один или более), микросхемы ОЗУ, ПЗУ, контроллеры внешних устройств и различные дополнительные элементы. Все эти элементы соединяются одно(двух) или многослойным печатным монтажом. Конструкция получается очень сложной. Поэтому наладка и поиск неисправностей традиционными методами (с помощью тестера, осциллографа и т.п.) сильно затруднены. Для облегчения этих работ такие платы снабжаются диагностическими разъемами. На этот разъем подключается специальное устройство, с помощью которого можно быстро найти неисправность. Такие устройства, называемые отладочными (или наладочными) стендами, обычно состоят из ПК и согласующего модуля для подключения к диагностическому разъему. На ПК отладочного стенда запускаются программы, которые управляют работой налаживаемой одноплатной ЭВМ. В результате проводимых проверок наладчик получает сведения о работоспособности проверяемой ЭВМ, и в случае обнаружения неисправности, подобрав соответствующую тестовую программу, может локализовать неисправный узел и отремонтировать его.

На участке наладки и контроля каждый наладчик имеет в своем распоряжении такой стенд. На производстве микро-ЭВМ УКНЦ (типичная одноплатная микро-ЭВМ) используется наладочный стенд, также выполненный на основе УКНЦ. Микро-ЭВМ стенда снабжена флоппи-дисководом и дискетами с тестовым матобеспечением.

Получив новую плату, подлежащую проверке, наладчик подключает ее к своему стенду через модуль сопряжения и запускает планирующую программу. На экране микро-ЭВМ стенда появляется меню для выбора тестов, из которого наладчик должен выбрать соответствующий тест. Поскольку он работает с новой, еще не проверенной платой, то запускает комплексный тест. Если комплексный тест проходит без сбоев, то считается, что проверяемая микро-ЭВМ исправна. Если же на экране по-

являются какие-либо сообщения об ошибках, например:

Ошибка ОЗУ ЦП 177777 177773

то наладчик выбирает более подробный тест, соответствующий обнаруженной ошибке. В нашем случае он запускает тест ОЗУ. Тест ОЗУ в свою очередь выдает следующее, более подробное сообщение об ошибке. Получив эту информацию, наладчик по электрической схеме определяет неисправную микросхему, заменяет ее и вновь запускает тест ОЗУ. Получив на дисплее сообщение:

Тест ОЗУ
Конец прохода. Ошибок 000000

наладчик снова запускает комплексный тест и по его успешному завершению переходит к работе со следующей платой. Рассмотрим теперь, какие преимущества может получить наладчик, если стендовый ПК будет подключен к системе виртуального доступа.

Вся система состоит из шести периферийных стендовых ПК УКНЦ, звездообразно соединенных с ЭВМ ДВК-3 (МС 0507.02). Периферийные УКНЦ не имеют никаких внешних устройств, кроме модуля сопряжения для подключения тестируемых микро-ЭВМ. Головная ЭВМ снабжена накопителем на винчестер-диске (10-20 Мб), накопителями на флоппи-дисках и устройством печати. На винчестере головной ЭВМ расположено тестовое матобеспечение, представляющее собой базу знаний и набор тестовых программ. База знаний — это программа, позволяющая логическим путем определять и локализовывать неисправности микро-ЭВМ. Естественно объем такой базы знаний достаточно велик и разместить ее на флоппи-дисках не представляется возможным, в то время как на винчестер-диске головной ЭВМ для нее места вполне достаточно.

При такой системе наладчик, после подключения проверяемой микро-ЭВМ, просто запускает головную программу базы знаний, которая в свою очередь запустит комплексный тест. Получив результаты комплексного теста, программа либо информирует наладчика об исправности платы, либо выбирает соответствующий тест для неисправного узла. В конце концов наладчик получит на дисплее фрагменты принципиальной и монтажной схемы и указание на то, какой элемент требует замены. Следовательно значительно повышается производительность труда наладчика и снижаются требования к его квалификации.

Кроме того, описываемая система содержит статистическую подсистему, которая фиксирует обнаруженные неисправности. Собранная информация передается затем в конструкторское бюро и отдел главного технолога для анализа и последующего улучшения конструкции выпускаемых микро-ЭВМ.

Все изменения в конструкции микро-ЭВМ легко вносятся в базу знаний головной ЭВМ.

Еще один пример использования системы виртуального доступа, на этот раз из области высшего образования. В ВУЗах, готовящих инженеров-технологов химических и микробиологических производств такая система с успехом может быть применена для организации лабораторного практикума по моделированию технологических процессов в масштабе реального времени или в ускоренном режиме.

Литература

1. В мире персональных ЭВМ. — М.: Радио и связь, 1988. — № 2. — С. 131 — 135.
2. Информатика и образование. — М.: Педагогика, 1989. — № 2 — С. 111.
3. Микропроцессорные средства и системы. — М.: 1988 — № 2 — С.53 -55.

Сети ЭВМ

Следует уточнить, что речь здесь будет вестись о так называемых локальных вычислительных сетях и микросетях, особенностью которых является то, что они размещены на ограниченной территории, а микросети даже в одном помещении, под контролем одной организации, но с возможностью выхода во внешнюю среду (в региональные или национальные вычислительные сети).

В общем виде сети ЭВМ состоят из набора различных устройств: ЭВМ, персональных ЭВМ, терминалов, устройств печати, файловых процессоров и т.д., соединенных между собой каналами связи. Каналы связи представляют собой витые пары, коаксиальные кабели, световоды и т.д. Каждое устройство подключается к каналу с помощью соответствующего контроллера.

Одна из основных отличительных особенностей сетей состоит в том, что каждое устройство, подключенное к сети, имеет возможность взаимодействия с любым другим устройством этой сети.

Сети характеризуются в первую очередь своей топологией. Наибольшее распростронение получили сети с кольцевой и шинной топологией (см.рис. 1 б,в). Кроме этого существуют также сети со звездообразной топологией (рис.1 а), древовидной (рис.2 а) и многосвязные сети (рис. 2б).

Топология сети во многом определяет надежность сети, сложность сетевых контроллеров, сетевого матобеспечения и соответственно сервиса, предоставляемого пользователю, и в конечном итоге стоимость сети. Стоимость сети также зависит от используемой среды передачи и в то же время от среды передачи зависит максимально возможная скорость передачи информации по сети и, следовательно, ее пропускная способность. Самые дешевые сети используют витую пару в качестве среды передачи, однако скорость передачи данных по витой паре не превышает 1 Мбит/с. Коаксиальный кабель более дорог, но и скорость передачи данных при его использовании достигает 10-15 Мбит/с. Самой дорогой средой передачи являются оптические световоды, скорость передачи данных по ним может достигать сотен Мбит/с. В локальных и микро-сетях ЭВМ обычно используются витые пары или коаксиальные кабели.

Сети ЭВМ имеют еще одну очень важную характеристику, во многом определяющую пропускную способность сети, ее надежность и стоимость сетевых контроллеров. Это способ доступа, который используется для передачи и приема информации по сети. Рассмотрим некоторые, наиболее часто применяемые способы доступа для сетей с кольцевой и шинной топологией.

При передаче информации по каналу связи сети используются так называемые информационные пакеты. Информационный пакет, как правило, включает преамбулу (т.е.набор синхронизирующих битов, которые дают возможность принимающей стороне приготовиться к приему), адресную часть, в которую записываются адреса передатчика и приемника, служебную вспомогательную часть, информационную зону, в которой записывается передаваемая информация, и контрольную сумму, с помощью которой проверяется правильность приема пакета (рис.3). Размер каждого поля в пакете варьируется в зависимости от параметров сети, например, адресное поле, длиной в три бита позволит адресоваться только к 8 различным устройствам, а адресное поле в 16 бит позволит адресоваться к 2-16 устройствам. Размер информационой зоны также может варьироваться. Чем больше размер информационной зоны, тем лучше соотношение объемов полезной и служебной информации в пакете. Однако при большом уровне помех в сети предпочтительнее короткие информационные поля, так как в этом случае снижается вероятность искажения информации. Для передачи информационного пакета абоненту устрой-

Рис.3. Сетевой информационный пакет

ство — контроллер должно каким-то образом передать его по каналу связи. В кольцевых сетях для этого чаще всего используется маркерный доступ. Его суть заключается в том, что по кольцу непрерывно циркулирует маркер — определенная последовательность бит. ЭВМ, которой требуется передать информацию, дожидается прихода маркера, определенным образом помечает его и выпускает вслед за маркером в кольцо свой информационый пакет с соответствующим образом установленными адресными полями. Все последующие машины, обнаружив помеченный маркер, читают адресную информацию пакета для проверки, не им ли направлен данный пакет. Если у какой-либо машины имеется готовое к передаче сообщение, то она пропускает данный пакет с помеченным маркером и ждет прихода непомеченного маркера. Наконец маркер достигает той ЭВМ, которой было направлено сообщение. Она читает это сообщение, устанавливает бит подтверждения приема без ошибки (проверив контрольную сумму) в служебном поле пакета и передает пакет дальше. Наконец пакет приходит к той ЭВМ, которая его послала. Она читает пакет, удаляет его из кольца и генерирует непомеченный маркер. Если она затем обнаружит, что в пакете не установлен бит приема сообщения или установлен бит приема с ошибкой, то она повторит передачу после того, как снова примет непомеченный маркер. Следовательно при таком методе доступа каждая ЭВМ равноправна и имеет право на посылку пакета информации по получении непомеченного маркера. Кроме того, каждая ЭВМ сети следит за проходящими через нее пакетами и удаляет все, проходящие во-второй раз (такие ситуации возникают, если передавшая пакет ЭВМ не удалит его из кольца), а также генерирует маркер специального вида, если в течение определенного промежутка времени не получает помеченный или непомеченный маркер, т.е. при потере маркера. В маркере специального вида ЭВМ записывает свой номер. Поскольку при потере маркера такую операцию выполняют все ЭВМ сети, то в кольце появляется множество маркеров с записанными в них номерами ЭВМ. Каждая ЭВМ при проходе через нее такого маркера удаляет его из кольца, если номер, записанный в нем меньше ее собственного номера. В итоге в кольце остается всего один маркер с наибольшим номером, который при очередном проходе через ЭВМ с этим номером преобразуется в обычный непомеченный маркер. Так поддерживается постоянное присутствие маркера в кольце вне зависимости от числа включенных ЭВМ и их номеров.

Следует отметить, что описанный метод отличается довольно высокими накладными расходами, т.е. для передачи каждого информационного пакета ему необходимо совершить полную циркуляцию по сети. Для увеличения производительности используются различные методы, например следующий. При прохождении маркера через очередную ЭВМ, она может передать информацию даже в том случае, если к маркеру

уже пристыкован информационный пакет. Для этого ЭВМ записывает свой пакет в конец уже имеющегося пакета. Последующие ЭВМ просматривают весь набор пакетов и удаляют из набора пакет, адресованный им, а на его место вставляют укороченный пакет, содержащий только служебную информацию, подтверждающую прием. ЭВМ — отправитель в свою очередь, получив эту информацию удаляет ее из "хвоста" за маркером.

В шинных сетях наибольшее распространение получил метод множественного доступа с контролем несущей и обнаружением конфликтов (CSMA/CD). Его суть заключается в следующем. В каждый момент времени любая ЭВМ сети может начать передачу своего информационного пакета, если в это время канал свободен. При передаче канал связи прослушивается передающей ЭВМ, и в случае искажения сигнала передача немедленно прерывается и включается отсчет тайм-аута. По истечении тайм-аута попытка передачи может быть повторена. Искажения сигнала при передаче, кроме помех, могут быть вызваны тем, что после начала передачи существует какой-то интервал времени, пока сигнал достигнет всех ЭВМ сети, и они обнаружат занятость линии. Если же за этим промежутком времени какая-либо ЭВМ тоже начнет передачу, то возникнет так называемая коллизия, которая будет обнаружена обеими передающими ЭВМ, и передача будет немедленно прекращена. Для избежания повторных коллизий величина тайм-аута в контроллерах сети такого типа выбирается по методу случайного числа. Пропускная способность таких сетей во многом зависит от нагрузки на сеть, так как при большой нагрузке значительно возрастает число коллизий и, следовательно, падает пропускная способность. Во избежание таких ситуаций скорость передачи выбирается таким образом, чтобы обеспечить загрузку сети не более чем на 50-65%.

Какой же сервис могут предоставить сети своим пользователям и в каких случаях применение сетей наиболее оптимально. В качестве примера можно рассмотреть кольцевую локальную сеть КУВТ-88, предназначенную для применения в учебных целях. В состав КУВТ-88 входит головная ЭВМ УКНЦ, оборудованная флоппи-дисководом и устройством печати, и 12 ЭВМ УКНЦ, не имеющая никаких периферийных устройств, кроме контроллера кольцевой локальной сети. Все эти ЭВМ связаны с помощью витой пары проводов в кольцо и размещены в пределах учебного класса. В сети используется маркерный метод доступа. Головная ЭВМ установлена на рабочем месте преподавателя. С головной ЭВМ можно осуществить загрузку программ во все периферийные ЭВМ. При работе на периферийных ЭВМ возможно использование прикладной программы, загруженной в эти ЭВМ, данных, расположенных на дисках головной ЭВМ, распечатки текстов на принтере головной ЭВМ. Преподаватель может контролировать и разрешать или запрещать эти действия с пульта головной ЭВМ. Описанная система не отвечает в полной мере требованиям, предъявляемым к сетям, но благодаря своей простоте и дешевизне находит широкое применение.

Одной из самых популярных локальных сетей в настоящее время является сеть Ethernet, разработанная консорциумом из трех фирм: Digital Equipment Corporation, Rank Xerox и Intel. Повсеместное внедрение этой сети сдерживается пока ее высокой стоимостью.

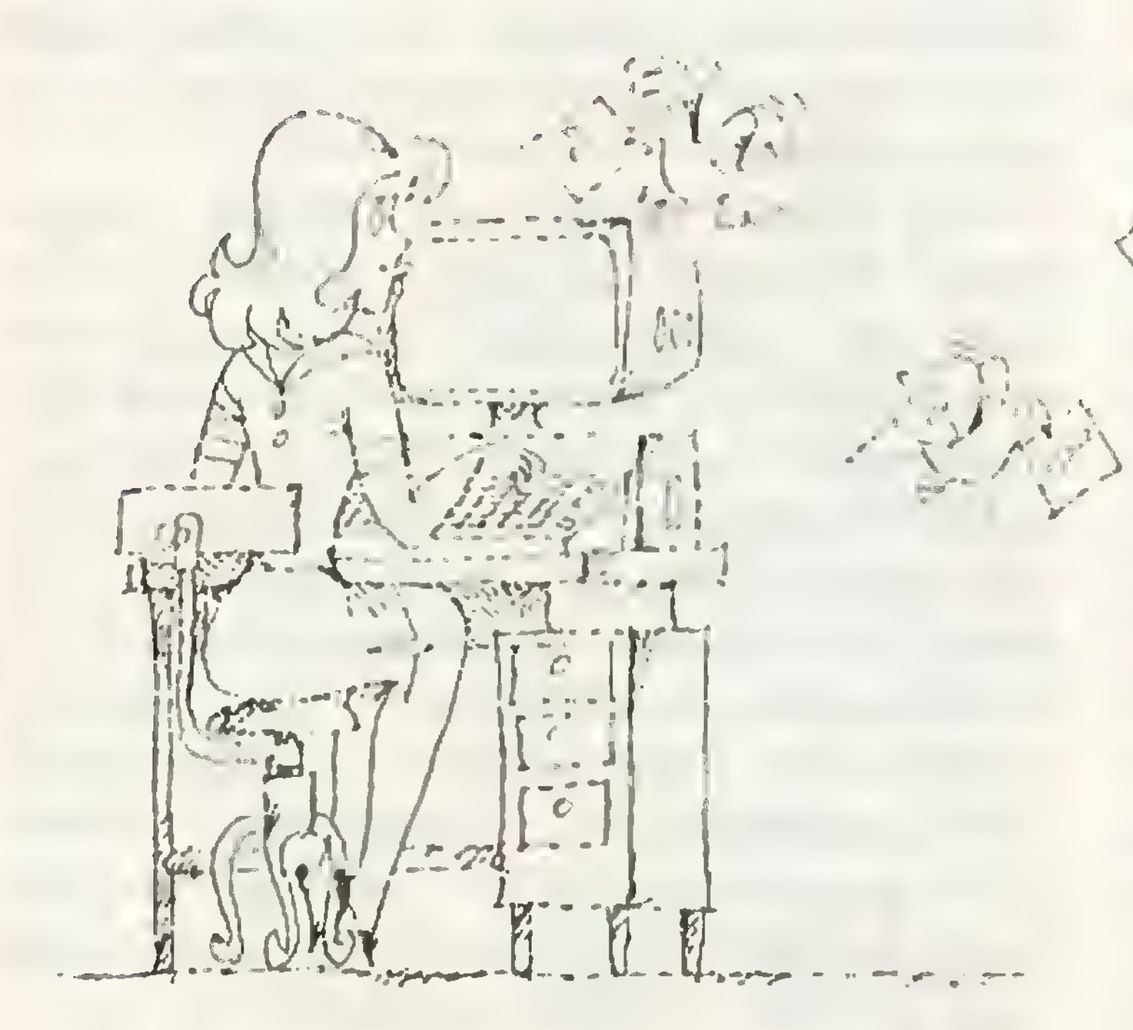
На базе локальных сетей, подобных Ethernet, строятся системы обработки информации в учреждениях и различных конторах, системы управления технологическими процессами (АСУТП), автоматизированные системы научных исследований (АСНИ), системы автоматизированного проектирования (САПР) и др. Следует отметить, что перечисленные автоматизированные комплексы могут быть реализованы и с помощью других систем коллективного пользования, о чем свидетельствует тот факт,

что такие комплексы создавались и до появления ПК и сетей на их основе. Однако есть целый ряд систем, появление которых стало возможным лишь после разработки компьютерных сетей на основе ПК. К ним относятся: банковские системы, не только ведущие счета клиентов, но и выполняющие оплату счетов за покупки на основе магнитных кредитных карточек; электронная почта; быстро входящие в моду компьютерные конференции и симпозиумы и.т.д.

Кратко остановимся на некоторых из этих возможностей (более подробно с применением ЛВС в различных сферах деятельности можно ознакомиться в [1] и [2]).

Одна из них — электронная почта — новейшиее средство пересылки и обработки текстов документов. Исследования, проведенные немецкими учеными [2] показывают, что электронная почта обеспечивает передачу текста в 25 раз быстрее и в 10 раз дешевле, чем телекс. При реализации электронной почты внутри одной фирмы и одного здания ее техническую основу составляет локальная вычислительная сеть и передача информации осуществляется по витой паре или коаксиальному кабелю. При передаче информации по отделениям фирмы, территориально удаленным друг от друга, используются телефонные каналы. Информационный банк данных, выполняющий роль почтового отделения, реализует такие функции, как получение и хранение корреспонденции, ее регистрацию и сортировку, отправку пользователю уведомления о приходе корреспонденции и т.д. Электронное почтовое отделение может выполнять и дополнительные функции, такие как защита корреспонденции от несанкционированного доступа копирование и размножение документов при рассылке идентичной корреспонденции по нескольким адресам и т.д.

В качестве еще одного примера можно привести использование локальной сети для создания мультипликационных фильмов. Каждый художник-мультипликатор, сидя за персональным компьютером с помощью пакета мультипликационной графики отрабатывает образ своего героя: его внешний облик, детали одежды, фазы движения. При этом художник как бы работает со своим актером, репетируя роль. Так, если бы кто-нибудь вздумал снять новый вариант "Белоснежки", то художникам представилась бы возможность создать свое собственное видение образов "простака", "застенчивого", "сердитого", Белоснежки и т.д. А далее, в соот-



ветствии со сценарием, программируется динамика их движения друг относительно друга, фон, на котором действуют персонажи и т.п.

Литература

1. Чоговадзе Г.Г. Персональные компьютеры. — М.: Финансы и статистика, 1989. — 208 с.
2. Блэк Ю. Сети ЭВМ. — М.: Мир, 1990. — 510 с.
3. Васильев Г.П. и др. Программное обеспечение неоднородных распределенных систем. — М.: Финансы и статистика, 1986. — 160 с.
4. Cheong V.E., Hirscheim R.A. Local Area Networks, John Wiley & Sons, Chichester. — 1986. — 190 p.
5. Ги К. Введение в локальные вычислительные сети. — М.: Радио и связь, 1986. — 176 с.

Проблема выбора

Ознакомив в общих чертах читателя с многотерминальными системами, системами виртуального доступа и сетями ЭВМ, в заключение дадим некоторые рекомендации по выбору типа системы, соответствующей задаче, стоящей перед пользователем.

Сначала несколько слов о цене. Почти во всех случаях цена многотерминальной системы будет ниже цены системы виртуального доступа или сети сравнимой мощности. Только в отдельных случаях при применении очень сложных "интеллектуальных" терминалов и мощной центральной ЭВМ с развитой периферией цена многотерминальной системы может превысить цену аппаратуры сетей. Однако даже в этом случае речь идет о цене аппаратного, а не программного обеспечения. Программное обеспечение многотерминальных систем значительно проще и дешевле програмного обеспечения систем виртуального доступа и сетей. Эксплуатация многотерминальных систем не требует такой высокой квалификации обслуживающего персонала электронщиков и программистов, как систем виртуального доступа и сети.

В то же время при использовании в качестве центральной ЭВМ многотерминальной системы средних или больших ЭВМ и внешних запоминающих устройств большой емкости для их установки требуется специально оборудованное помещение с кондиционированием воздуха и фальшполом, со специальной шиной питания. Энергопотребление такой системы может быть достаточно большим.

Простейшие системы виртуального доступа, как и простейшие сети, могут иметь очень низкую стоимость, однако их простота означает очень

узкую специализацию и крайне ограниченный набор функций. Удельная стоимость, т.е. стоимость единицы вычислительной мощности систем виртуального доступа и сетей ЭВМ, как правило, ниже, чем у многотерминальных систем, однако этот показатель не критичен для сетей ЭВМ, а для систем виртуального доступа он играет серьезную роль только при решении задач, требующих значительных вычислительных ресурсов.

Квалификация обслуживающего персонала для систем виртуального доступа и особенно для сетей ЭВМ должна быть очень высокой. Чаще всего это касается этапов монтажа и пуска в эксплуатацию. При дальнейшей нормальной эксплуатации систем виртуального доступа и сетей ЭВМ основные трудности возникают со стороны матобеспечения, а не аппаратуры. Стоимость системного матобеспечения сетей и систем виртуального доступа выше стоимости системного матобеспечения многотерминальных систем, в то время как стоимость прикладного матобеспечения систем виртуального доступа практически не отличается от стоимости прикладного матобеспечения многотерминальных систем. Стоимость как системного, так и прикладного матобеспечения сетей ЭВМ обычно значительно превышает стоимость аналогичного матобеспечения систем виртуального доступа и многотерминальных систем.

Теперь по порядку об областях применения. Многотерминальные системы в первую очередь позволяют простейшим путем организовать такую работу пользователей системы, при которой все они находятся под жестким контролем управляющей программы или менеджера системы. При этом легко организуется как управление действиями пользователя, так и управление распределением ресурсов системы. Эффективное использование возможностей центральной ЭВМ достигается при решении на многотерминальной системе задач, требующих больших объемов ввода/вывода информации. Хорошо организуются на многотерминальных системах варианты, требующие частого взимодействия пользователей друг с другом, системы реального времени. В то же время на многотерминальных системах нерационально решать задачи, связанные с большим объемом вычислений в мультизадачном режиме. Очень хорошие с экономической точки зрения результаты могут быть получены в случае применения многотерминальных систем в узкоспециализированной области с использованием специализированных терминалов. В тех случаях, когда на рабочих местах пользователей требуется установка оборудования для работы с графикой, стоимость многотерминальной системы становится сопоставимой со стоимостью сети из простых ПК с графическими дисплеями. В этом случае, если планируется дальнейшее расширение и модификация системы предпочтительнее использовать сети, так как они дают более гибкие возможности для развития. Системы виртуального доступа имеют довольно узкую область применения. Они используются в основном в тех случаях, когда требуются большие вычислительные мощности при низкой цене системы за счет минимизации количества дисковых периферийных устройств. Такие условия характерны для процессов обучения (машинное моделирование, курсовое проектирование) или решения сложных задач в многопроцессорном режиме. Кроме того, с помощью системы виртуального доступа удачно решаются производственные задачи, подобно описанной во второй главе. Если же при выборе системы подразумевается какой-то спектр рашаемых задач с неизвестными заранее характеристиками, то целесообразно будет использование сети ЭВМ с подсистемой виртуального доступа.

Собственно использование сети ЭВМ — это в основном вопрос финансовых возможностей, поскольку, несмотря на постоянное снижение цен, сети все же еще остаются дорогим удовольствием. С помощью сетей ЭВМ возможно решение широко-

го спектра задач, начиная от организации безбумажной технологии в офисе и кончая системой управления спутниками. Однако, как и во всех случаях, здесь следует рекомендовать принцип разумной достаточности. Например, для организации складского учета и кассовой системы небольшого магазина гораздо эффективнее использовать многотерминальную систему, а не сеть, в то время как для такой же системы большого универмага (типа ГУМа) многотерминальная система экономически менее выгодна, чем сеть ЭВМ, так как для многотерминальной системы потребуется очень мощная центральная ЭВМ и такая же резервная, поскольку выход ее из строя приведет к катастрофе. Выход же из строя отдельных ЭВМ в сети не вызовет тяжелых последствий.

Заключение

В заключение отметим некоторые основные тенденции, которые намечились в настоящее время в развитии систем коллективного пользования.

1. Создание в структуре организации крупных вычислительных комплексов на базе мини-супер и супер-ЭВМ, предназначенных для использования в научно-исследовательских и конструкторских разработках. Ярким примером такой системы является комплекс Эймского центра NASA стоимостью 120 млн. долларов, предназначенный для проектирования авиационно-космических аппаратов и для моделирования сложных процессов. Основу комплекса составляет супер-ЭВМ Cray-2. Кроме того в комплекс входят две ЭВМ Amdahl 5840 для управления внешними устройствами, четыре ЭВМ VAX-11/780 фирмы DEC, 27 графических устройств iRIS-2500 turbo и 11 графических устройств iRIS-3030. В комплексе используются также 279 быстродействующих дисковых запоминающих устройств. На каждом из 30 автоматизированных рабочих мест, подключенных к центральному процессору Cray-2 через устройства связи Hyperchannel со скоростью передачи 50 Мбит/с, имеется 24-цветовой дисплей с растром 1024x1024 элементов, позволяющий видеть трехмерную картинку обтекания тела потоком в динамике и осуществлять вращение. Для комплекса построено двухэтажное здание, площадью 8.3 тыс. кв. м.

2. Создание национальных центров, оснащенных значительными вычислительными мощностями и предоставление доступа к этим центрам организациям и индивидуальным пользователям через национальные сети. В 1985-1986 гг. в США были созданы пять таких центров, оснащенных супер-ЭВМ. Пользователи получают доступ к любому из этих центров через сети NSFNET, TELENET, ARPANET, BITNET, CSNET. В настоящее время проводится периодическое дооснащение и переоснащение этих центров новейшими моделями супер-ЭВМ для обеспечения возрастающих потребностей пользователей.

3. Использование сверхбыстродействующих сетей в системах распределенной обработки информации. Так например, в системе Crystal двадцать узловых процессоров VAX-11/750 соединены в кольцевую сеть с маркерным доступом и скоростью передачи 80 Мбит/с. Такая конфигурация позволяет использовать систему многим пользователям для различных научно-исследовательских проектов, причем вычислительные мощности, выделяемые на каждый проект, распределяются в соответствии с потребностями проекта.

Ограниченный объем работы не дает возможности более подробно остановиться на перспективах развития вычислительных систем, поэтому читателям, желающим более подробно ознакомиться с этим, мы можем порекомендовать следующую литературу:

"Вычислительная техника за рубежом в 1986-1987 гг."/Под.ред.Г.Г.Рябова. М.: ИТМ и ВТ АН СССР, 1987. — 331 с.;

"Вычислительная техника за рубежом в 1987-1988 гг."/Под ред Г.Г.Рябова. М.: ИТМ и ВТ АН СССР, 1988. — 316 с.

БК ЗА РОГА

Эта статья посвящена тому, как получить доступ к встроенным возможностям БЕЙСИКа из программ в кодах и на языке АССЕМБЛЕРа. Все сказанное будет относиться к версии БЕЙСИКа "Вильнюс 1986.07.24", поставляемой заводом-изготовителем вместе с БК-0010.01.

Все приведенные здесь тексты программ на языке АССЕМБЛЕРа написаны в формате входного языка для системы "Микро". Однако, если эта система у вас отсутствует, то не отчаивайтесь: текст программ без особого труда можно переделать для вашей любимой системы ("Мираж", "Shura", "ОС-0010", "ОТЛ" и т.д.) или просто ввести в кодовом варианте с помощью тестового монитора.

В.В.Авсеев,
А.В.Авсеев

ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ БЕЙСИКА В ПРОГРАММАХ НА ЯЗЫКЕ АССЕМБЛЕРА

Пользователи БК-0010 в модели БК-0010.01 получили новую клавиатуру и "зашитый" в ПЗУ транслятор языка БЕЙСИК, о котором говорилось, что он соответствует популярному стандарту MSX. Кроме того, в руководстве заявлялось о существовании в дополнительном внешнем блоке ПЗУ интерпретатора языка ФОКАЛ. Таким образом, пользователь получал возможность работать сразу с двумя языками высокого уровня на своем компьютере.

Когда первые БК-0010 появились в магазине "Электроника", авторы этой статьи, не задумываясь, приобрели его (тогда это можно было сделать без всякой очереди). Но вскоре радость приобретения сменилась некоторым разочарованием. Все поиски интерпретатора ФОКАЛа в дополнительном блоке ПЗУ были тщетными, клавиатура оказалась по качеству не на много лучше пленочной, особенно бросался в глаза известный теперь всем пользователям БК-0010.01 "дребезг" клавиатуры. Но больше всего авторов поразило, что обещанных в руководстве демонстрационных программ на БЕЙСИКе не оказалось, вместо них были записаны демонстрационные программы на ФОКАЛе (видимо, на заводе-изготовителе тоже есть шутники). Не с хорошей стороны проявил себя и транслятор БЕЙСИКа. Авторы сразу столкнулись с проблемой нехватки памяти, непонятными зависаниями, замедлением работы с символьными переменными, ненадежной работой некоторых функций, неправильной реализацией многомерных массивов и т.д.

Правда, потом оказалось, что с "дребезгом" клавиатуры довольно легко можно справиться. Без ФОКАЛа в принципе можно обойтись. Но недостатки БЕЙСИКа исправить без замены ПЗУ никак нельзя. Оставался один выход: переходить к программированию на языке АССЕМБЛЕРа. Но в таком случае возникает много трудностей из-за отсутствия возможностей, предоставляемых пользователю БЕЙСИКом. Конечно, написать подпрограммы, реализующие сложные графические операторы БЕЙСИКа (рисование окружности, закрашивание замкнутой области), можно и на языке АССЕМБЛЕРа. Но всегда ли оправдано изобретать велосипед? Иногда оказывается полезным использовать уже заложенные в ПЗУ БК-0010.01 возможности.

Поиск адресов подпрограмм

Большинство трансляторов БЕЙСИКа является интерпретаторами. Но на ПЭВМ БК-0010.01 реализован компи-

лятор БЕЙСИКа: исходная программа после ее запуска командой RUN преобразуется в памяти ЭВМ в прямой шитый код (детальное определение шитого кода см., например,: Баранов С.Н., Новдрунев Н.Р. Язык ФОРТ и его реализации. — М: Машиностроение, 1988. — С.43). Прямой шитый код представляет собой последовательность адресов подпрограмм, констант и адресов переменных. Один оператор БЕЙСИКа реализуется с помощью одной или нескольких таких подпрограмм.

Узнать адреса подпрограмм, реализующих тот или иной оператор довольно просто. Авторы уже останавливались на этом вопросе в журнале "Информатика и образование" [1]. В этом же номере приведен почти полный список адресов подпрограмм и операторов БЕЙСИКа.

Приведенная далее программа на БЕЙСИКе выдает на экран шитый код, соответствующий вашим операторам, записанным в первых строках этой программы. Все числа выводятся в восьмеричной системе исчисления. Программа использует ячейку &O2002, в которой хранится адрес последнего байта исходного текста программы на БЕЙСИКе. Шитый код будет начинаться со следующего большего четного адреса. Предполагается, что среди ваших операторов, записанных в первых строках программы, отсутствует CLS, так как именно до этого оператора будет выводиться шитый код. Подпрограмма, реализующая оператор CLS, находится по адресу &O156776.

```
10 REM Программа запускается
20 REM Командой "RUN 100"
30 REM Здесь должны находиться
40 REM ваши операторы
50          ...
100 CLS
110 N% = PEEK(&O2002)
120 IF N% MOD 2% = 1% THEN
    N% = N% + 1%
130 IF PEEK(N%) = &O156776 THEN
    STOP
140 ? OCT$(PEEK(N%))
150 N% = N% + 2%
160 GOTO 130
170 END
```

В приведенной ниже таблице каждому оператору языка, записанному в общей форме, сопоставляется соответствующий ему шитый код — содержимое последовательно расположенных в ОЗУ микро ЭВМ 16-разрядных слов. В шитом коде, кроме адресов подпрограмм, могут располагаться адреса переменных и другие данные. Типы параметров, приводимых в таблице, соответствуют стандарту Бейсик-MSX и обозначаются: # — вещественное двойной точности; ! — вещественное одинарной точности; % — целое число. Параметры, помещенные в первой части таблицы в скобках, означают, что их необходимо переслать на стек в порядке перечисления. На вершину стека указывает регистр R6(SP) процессора микро-ЭВМ. Переслать параметры на стек можно при помощи вспомогательных подпрограмм БЕЙСИКа, также перечисленных в таблице. В приведенной таблице в отличие от таблицы в журнале "Информатика и образование" занесены лишь наиболее употребительные операторы и системные адреса:

Операторы

PRINT A%	(A%) 157070
PRINT AI	(AI) 157014
PRINT A#	(A#) 157022
INPUT X1,X2,...,Xn	160634, 157150, адрес следующей команды, список адресов X1,...,Xn
CLS	156776
CIRCLE(A%,B%),C%,D%	(A%,B%,C%, D%) 125334, 126724,126742, 127122,127324
PAINT(A%,B%), C%, D%	(A%, B%, C%) 125334, (D%), 125334, 125642
DRAW A$	(адрес A$, длина A$) 131412

Отношения и логические операции

A%<B%	(A%, B%)	155172,	155452
A#<B#	(A#, B#)	177034,	155452
A%>B%	(A%, B%)	155172,	155456
A#>B#	(A#, B#)	177034,	155456
A%>=B%	(A%, B%)	155172,	155446
A#>=B#	(A#, B#)	177034,	155446
A%<=B%	(A%, B%)	155172,	155442
A#<=B#	(A#, B#)	177034,	155442
A%<>B%	(A%, B%)	155172,	155436
A#<>B#	(A#, B#)	177034,	155436
NOT A%	(A%)	162306	
A% AND B%	(A%, B%)	162316	
A% OR B%	(A%, B%)	162312	
A% XOR B%	(A%, B%)	162324	
A% EQV B%	(A%, B%)	162340	
A% IMP B%	(A%, B%)	162344	
A%=B%	(A%, B%)	155172,	155430
A#=B#	(A#, B#)	177034,	155430

Функции

SQR(X#)	(X#)	171350
SIN(X#)	(X#)	173614
COS(X#)	(X#)	173566
TAN(X#)	(X#)	174306
ATN(X#)	(X#)	174434
EXP(X#)	(X#)	171762
PI		167102
LOG(X#)	(X#)	173052
ABS(X#)	(X#)	166566
FIX(X#)	(X#)	176212
INT(X#)	(X#)	176340
SGN(X%)	(X%)	166572
SGN(XI)	(XI)	166624
SGN(X#)	(X#)	166614
RND(X#)	(X#)	175176
CINT(X#)	(X#)	160760
CSNG(X#)	(X#)	162020
CDBL(AI)	(AI)	166742
CDBL(A%)	(A%)	166646
VAL(A$)	(адрес A$, длина A$)	161550
STR$(A%)	(A%)	161610 (JSR PC,140536)
STR$(A#)	(A#)	161602 (JSR PC, 164710)
STR$(AI)	(AI)	161574 (JSR PC, 164664)
OCT$(A%)	(A%)	161624
BIN$(A%)	(A%)	161672
HEX$(A%)	(A%)	161734

Адреса вспомогательных подпрограмм

156160, A% - запись слова A% на стек
156156, AI - запись двух слов на стек
156152, A# - запись четырех слов на стек
156232, A% - пересылка слова по адресу A% на стек
156214, A% - пересылка двух слов по адресу A% на стек
156170, A% - пересылка четырех слов по адресу A% на стек
156350, A% - запись слова со стека по адресу A%
156330, A% - запись двух слов со стека по адресу A%
156334, A% - запись четырех слов со стека по адресу A%
166562 - изменение знака целого числа на стеке
156750 - изменение знака вещественного числа на стеке

Некоторые системные адреса БЕЙСИКа

2002 - ячейка, содержащая адрес конца текста программы
2004 - ячейка, содержащая адрес конца шитого кода программы
37400 - адрес буфера ввода-вывода с магнитофона
2422 - адрес буфера ввода с клавиатуры
3022 - адрес буфера преобразования в строку и вывода чисел на экран
120300 - адрес перезапуска БЕЙСИКа из монитора с сохранением программы
120234 - адрес подпрограммы обработки прерывания по нажатию клавиши "СТОП"
121052 - адрес вызова строкового редактора текстов Бейсика

Арифметические операции

A% + B%	(A%, B%)	162102
A# + B#	(A#, B#)	167144
A% - B%	(A%, B%)	162112
A# - B#	(A#, B#)	167124
A# / B#	(A#, B#)	170776
A% \ B%	(A%, B%)	162052
A% * B%	(A%, B%)	162116
A# * B#	(A#, B#)	170210

Заметим, что в таблице буквами A — D обозначены константы, при

этом константа A% занимает одно слово (два байта), A! — два слова, A# — четыре слова.

Некоторые неудобства при использовании операторов БЕЙСИКа из программ в кодах или на языке АССЕМБЛЕРа доставляет способ их вызова: для этого требуется записать в регистр R4 адрес слова ОЗУ, где расположен адрес подпрограммы, а затем выполнить команду JMP @(R4)+. Такова особенность компилятора БЕЙСИКа — передача управления от одной подпрограммы к другой осуществляется через регистр R4 по команде JMP @(R4)+, содержащейся в конце каждой подпрограммы. В связи с этим желательно весь список адресов подпрограмм (в вашей программе) записывать в ОЗУ подряд, тогда автоматически после одной подпрограммы будет вызвана на исполнение следующая. Многие подпрограммы БЕЙСИКа используют служебную область памяти и прерывание TRAP. Основная служебная область БЕЙСИКа начинается с адреса &O2000. Адреса &O2422 — &O3022 занимает буфер строкового редактора (его длина 256 байт), в самом конце этого буфера располагается маленькая область, используемая оператором PRINT для преобразования чисел в символьное представление. Начальный адрес этой области — &O3002. При использовании операторов ввода/вывода на внешнее устройство (магнитофон, локальная сеть) будет задействован буфер с начальным адресом &O37400. При использовании операторов БЕЙСИКа не желательно изменять содержимое памяти по адресам &O200 — &O3052, так как эти ячейки используются подпрограммами БЕЙСИКа для своих нужд. Область памяти &O1000 — &O2000 БЕЙСИК в принципе не использует, но во время его работы эта область задействована под системный стек. Если переместить начало стека на его "обычное" место (адреса &O736 — &O1000), то областью памяти &O1000 — &O2000 можно пользоваться без опасений. При использовании операторов ввода/вывода нужно обязательно обнулить ячейку по адресу &O2416, а перед вызовом арифметических операций также нужно обнулить ячейку &O2314.

Ниже приводится пример использования подпрограмм БЕЙСИКа.

TRAP — запросы БЕЙСИКа

В БЕЙСИКе, как и в ФОКАЛе, TRAP-запросы выполняют служебную роль. При этом запросы с номерами 0 — &O77 отведены для прерывания работы по ошибочной ситуации во время выполнения программ на БЕЙСИКе, а запросы с номерами больше

Адрес	Код команды		Команда АССЕМБЛЕРа
1000:	12737,146404,34		MOV #146404,@ #34
1006:	105737,40		TSTB @ #40
1012:	1003		BNE 1
1014:	12700,233		MOV #233,RO
1020:	104016		EMT 16
1022:	12704,1034	1:	MOV #PRG,R4
1026:	134		JMP @(R4) +
1030:	104006	EXT:	EMT 6
1032:	0		HALT
1034:	156156,175,170,156160	PRG:	. #156156. #175. #170. #156160
1044:	24,125376,126724		. #24. #125376. #126724
1052:	126742,127122,127334		. #126742. #127122. #127334
1060:	156156,175,170,125376		. #156156. #175. #170. #125376
1072:	125404,125642,1030		. #125404. #125642. #EXT
			.END

&O77 вызывают внутренние служебные подпрограммы БЕЙСИКа. Во время выполнения подпрограмм БЕЙСИКа могут возникнуть ошибочные ситуации: деление на нуль, переполнение разрядной сетки, недопустимое использование функции и т.д. Конечно, это приведет к выполнению TRAP-запроса с номером, равным номеру соответствующей ошибки в БЕЙСИКе. Если вы не напишите свой вариант "ловушки" для TRAP-запросов и не установите ее начальный адрес в ячейку &O34, использование подпрограмм БЕЙСИКа может привести к непредсказуемым последствиям. Поэтому мы приводим здесь свой вариант текста "ловушки" TRAP-запросов:

```
    MOV R0,-(SP)
    MOV R2,-(SP)
    MOV 4(SP),R0
    MOV 177776(R0),R2
    BIC #177440,R2
    CMP R2,#100
    BLO 1
    JMP @#146424
1 :  JMP ERR ; переход по ошибке
    ...
    ...
ERR:
```

После метки ERR: должна располагаться программа, реагирующая на возникновение ошибки. При этом в R2 будет храниться номер ошибки (список ошибок и их номера можно посмотреть в руководстве по БЕЙСИКу). В начале вашей программы, использующей подпрограммы БЕЙСИКа, запишите в ячейку &O34 адрес этой "ловушки".

Оригинальная идея использовать TRAP-прерывания по ошибке 5 принадлежит И. Николаеву (см. журнал "Информатика и образование" [2]). В приведенной там программе применяется перехват прерывания TRAP 5, возникающего при попытке выполнить команду LOAD в момент, когда в памяти уже есть текст некоторой программы. Вообще, использование TRAP-запросов для своих целей при вызове подпрограмм БЕЙСИКа может в некоторых случаях привести к зависаниям. Задействование же TRAP-запросов при работе компилятора вообще невозможно.

Арифметика чисел с плавающей запятой

БЕЙСИК-БК представляет более широкие возможности для работы с вещественными числами по сравнению с ФОКАЛом. Но механизм доступа к подпрограммам, реализующим плавающую арифметику, более неудобен. Он возможен либо командой JSR PC,<N>, где <N> — адрес подпрограммы, или командой JMP @(R4)+. Прерывание по резервной команде процессора в БЕЙСИКе вообще не задействовано (в отличие от ФОКАЛа), и реакцией на нее будет сообщение об ошибке с номером 52. Но использование встроенных возможностей арифметики БЕЙСИКа экономит память и время для написания новых подпрограмм. Примером использования плавающей арифметики может служить следующая программа. Она использует встроенные подпрограммы БЕЙСИКа: сложение, вычитание, деление и умножение числел, оператор PRINT A# и функцию VAL(A$).

```
        MOV #TRAP,@#34   ; УСТАНОВКА
                         АДРЕСА
                         TRAP-ЛОВУШКИ
        MOV #1000,SP     ; УСТАНОВКА
                         УКАЗАТЕЛЯ
                         СТЕКА
        CLR @#2416       ; ВЫВОД НА
                         ЭКРАН
        CLR @#2134
        TSTB @#40        ; УСТАНОВКА
                         ЦВЕТНОГО
                         РЕЖИМА
        BNE 1
        MOV #233,R0
        EMT 16
1:      MOV #ST2,R1      ; ВЫДАЕМ
                         НАЧАЛЬНУЮ
                         ЗАСТАВКУ
        JSR PC, OUS
CIC:    MOV #M1,EXT      ; ЗАПОМИНАЕМ
                         АДРЕС ВЫХОДА
M3:     MOV #ST3,R1      ; ВЫВОДИМ
                         ОПЕРАНД
        JSR PC,OUS
        JSR PC,@#121052
        MOV #VAL,R4
        JMP @(R4)+
M1:     MOV #M2,EXT      ; ЗАПОМИНАЕМ
                         АДРЕС ВЫХОДА
```

```
        BR M3
M2:     MOV #ST4,R1
        JSR PC,OUS
        EMT 6                   ; ВВОД
                                  ОПЕРАЦИИ
        EMT 16
        MOV #ST5,R1
        JSR PC,OUS
        CMPB RO,'+'
        BNE 1
        MOV #PR1,R4             ; СЛОЖЕНИЕ
                                  ЧИСЕЛ
        JMP @(R4)+
1:      CMPB RO,'-'
        BNE 2
        MOV #PR2,R4             ; ВЫЧИТАНИЕ
                                  ЧИСЕЛ
        JMP @(R4)+
2:      CMPB RO,'/'
        BNE 3
        MOV #PR3,R4             ; ДЕЛЕНИЕ ЧИСЕЛ
3:      JMP @(R4)+
        CMPB RO,'.'
        BNE 4
        MOV #PR4,R4             ; УМНОЖЕНИЕ
                                  ЧИСЕЛ
4:      JMP @(R4)+
        TRAP 0
TRP:    MOV #ST1,R1             ; БЛОК TRAP-
                                  ЛОВУШКИ
        JSR PC,OUS
        MOV #1000,SP
        JMP CIC
OUS:    CLR R2                  ; ПОДПРОГРАМ-
                                  МА ВЫВОДА
        EMT 20                  ; СТРОКИ НА
                                  ЭКРАН
        RTS PC
ST1:    .B:12.A:ОШИБКА ! ПОВТОРИТЕ ВВОД
        .B:12 E
ST2:    .B:14.A: НЕБОЛЬШОЙ КАЛЬКУЛЯТОР
        .B:12.E
ST3:    .B:12.A: ВВЕДИТЕ ОПЕРАНД:.E
ST4:    .B:12.A: ОПЕРАЦИЯ (-,+,*,/):.E
ST5:    .B:12.A:РЕЗУЛЬТАТ = .E
VAL:    .#156156.#2422.#400.#161550
EXT:    .E
PR1:    .#167144.#157022.#CIC
PR2:    .#167124.#157022.#CIC
PR3:    .#170776.#157022.#CIC
PR4:    .#170210.#157022.#CIC
        END
```

12737	4214	34	12706	1000
5037	2416	5037	2134	105737
40	1003	12700	233	104016
12701	4276	4767	166	12767
4076	344	12701	4326	4767
150	4737	121052	12704	4410
134	12767	4106	314	763
12701	4350	4767	116	104006
104016	12701	4374	4767	102
120027	53	1003	12704	4422
134	120027	55	1003	12704
4430	134	120027	57	1003
12704	4436	134	120027	52
1003	12704	4444	134	10400
12701	4242	4767	10	12706
1000	167	177612	5002	104020
207	47412	164773	45742	20101
20041	47760	52102	50117	52351
20105	41102	162117	5056	0
44014	161105	166117	175770	165117
45440	166101	45770	166365	52361
50117	12	41012	42502	164744
42524	47440	42760	40520	162110
72	47412	42760	40520	164743
24361	26053	26055	26052	24457
72	50012	175105	166365	52370
52101	75	156156	2422	400
161550	0	167144	157022	4046
167124	157022	4046	170776	157022
4046	170210	157022	4046	

Обратите внимание, что программа "калькулятор" использует для ввода строк с клавиатуры не стандартную функцию EMT 10, а встроенный редактор строк БЕЙСИКа. В отличие от EMT 10 этот редактор допускает довольно мощные возможности редактирования. Вызывается этот редактор командой JSR PC,@#121052. Текст вводится в буфер с начальным адресом &O2422, длина буфера — 256 байт. Заметим, что формат представления чисел с плавающей запятой в БЕЙСИКе отличен от формата, применяемого в ФОКАЛе БК. В вещественных числах одинарной точности в первом слове с 7-го по 14-й бит занимает порядок числа, старшие разряды мантиссы занимают с 0-го по 6-й бит этого же слова. Младшие разряды мантиссы располагаются в следующем слове. В вещественных числах двойной точности точность увеличивается за счет удлинения мантиссы. Надо заметить, что в БЕЙСИКе нет, например, арифметических операций над вещественными числами одинарной точности (точнее, нет таких подпрограмм). Перед выполнением арифметических операций над такими числами БЕЙСИК производит преобразование типа: CDBL(A!).

К вопросу об экономии памяти

Память БК стала объектом злых шуток и поговорок. Пожалуй, каждый пользователь БК сталкивается с про-

блемой экономии памяти. БЕЙСИК-БК является на редкость незкономным языком. Мало того что операторы в БЕЙСИКе довольно длинные, он их при этом никак не кодирует для хранения в памяти (подавляющее большинство трансляторов БЕЙСИКа кодируют операторы в специальный внутренний экономный код). Во время же компиляции память "поедается" под размещение шитого кода, который, кстати сказать, не совсем рационально реализован. Многие пользователи БК считают, что секрет экономии памяти заключен в уменьшении длины исходного текста программы. Но это не всегда так. Простейшим примером является цикл БЕЙСИКа:

```
1 FOR I = 1 TO 100
2 NEXT
```

Можно записать этот пример и по другому:

```
1 FOR I% = 1% TO 100%
2 NEXT I%
```

Первая программа по тексту короче второй. Однако если мы посмотрим после команды RUN объем свободной памяти (в первом и во втором случае), то окажется, что второй вариант занимает в памяти на 22 байта меньше. Причина этого в том, что в первой программе переменная I будет по умолчанию принята как вещественная двойной точности, а во второй мы принудительно указали, что переменная 1 является целой (экономия в байт). Это же относится и к константам 1 и 100 (еще 12 байт). Интересно отметить, что NEXT занимает в шитом коде места больше, чем NEXT I%. Здесь можно сэкономить на длине текста программы, но больше потерять за счет увеличения длины шитого кода. Поэтому никогда не используйте NEXT без явного указания имени переменной. Если возможно, то чаще используйте целые переменные и константы. Если вы используете целые или вещественные одинарной точности константы и переменные, то не забывайте явно указывать их тип. Но все-таки обычно сокращение длины текста программы сопровождается и уменьшением занимаемой памяти ПЭВМ.

Вот пример простой задержки в программе:

```
100 A$ = INKEY$
110 IF A$ = "" THEN 100
```

Этот пример можно сделать и таким:

```
100 IFINKEY$ = ""TH100
```

Во втором случае мы сэкономим 37 байт и за счет уменьшения длины текста и за счет сокращения шитого кода.

Авторы хотели бы предупредить, что при отладке программы "Калькулятор" в системе "Микро" перед перезапуском нужно перезагрузить ее с МЛ, так как используемые в программе подпрограммы БЕЙСИКа размещают свои переменные в области памяти &O2000 — &O3022 и портят при этом "Микро". Не пытайтесь также запускать приведенные в статье программы при подключенном внешнем блоке ПЗУ с тестовым монитором (и, может быть, ФОКАЛом), ведь при этом ими будут перекрыты области памяти, занимаемые подпрограммами БЕЙСИКа.

В заключение авторы хотели бы выразить благодарность студенту МАИ Александру Сергееву. Именно его письмо (Вычислительная техника и ее применение. — 1989. — № 9. — С.46-47) убедило нас, что данная статья нужна пользователям БК-0010.01.

Литература

1. Авсеев В.В., Авсеев А.В. Особенности транслятора с языка Бейсик для БК-0010.01// Информатика и образование. — 1990. — № 1. — С.42-46.

2. Николаев И. Нуждается в доработке // Информатика и образование. — 1989. — № 1. — С.101.

3. Осетинский Л.Г., Осетинский М.Г., Писаревский А.Н. ФОКАЛ для микро- и мини-компьютеров. — М.: Машиностроение, 1988.

4. Ларкин М. Экономия памяти БК-0010.01//Информатика и образование. — 1990. — № 3. — С.44-49.

В одном из номеров "Вычислительной техники" как-то прозвучал призыв к создателям Вильнюсской версии БЕЙСИКа рассказать о возможности использования подпрограмм БЕЙСИКа в программах пользователя, написанных в машинных кодах. Однако до сих пор (а прошел уже год) никто на этот призыв не откликнулся. По-видимому, исследованием БЕЙСИКа придется заниматься самим пользователям БК.

Д.Ю.Усенков
(Москва)

О МОНИТОРЕ И ВИЛЬНЮССКОЙ ВЕРСИИ БЕЙСИКА

Предлагаю вниманию читателей несколько своих наблюдений.

1. При нажатии клавиши на клавиатуре БК код, посылаемый этой клавишей в компьютер, зависит только от положения клавиши на клавиатуре. Все преобразования, приводящие к получению кодов символов (русские-латинские; заглавные-строчные; буквы-полуграфика и т.д.), происходят в программе-драйвере клавиатуры. Так, АР2 — переключатель, указывающий БК какой из двух векторов прерывания выбрать: @#60 или @#274. Клавиши РУС и ЛАТ обрабатываются драйвером по прерыванию @#60, управляя записью кода-маски в ячейку ОЗУ @#43_8 (ЛАТ-0; РУС-200_8). Клавиши ЗАГЛ и СТР работают без участия программы-драйвера, но, по сути, выполняют те же самые действия. После записи числа в ячейку @#43_8 все другие коды символов-букв устанавливаются по этому числу, как по маске, таким образом преобразуясь в символы русских или латинских букв в зависимости от режима клавиатуры. Основываясь на этом свойстве БК, я написал небольшую программу для БЕЙСИКа. Привожу здесь ее текст в восьмеричных кодах:

Адрес программы может быть любым. Обращение к ней производится через USR-функции. Программа написана так, что после ее загрузки с м/ф командой BLOAD "<имя>" R (обязательно с R!) оператор DEFUSR не требуется. Используется функция USR0. Если же вы хотите задействовать USR-функцию с другим номером, измените выделенные подчеркиванием коды с 2100 на 2102 (для USR1), 2104 (для USR2) и т.д. При обращении аргумент функции обязательно должен быть целым числом (типа %). Исходный режим до запуска программы должен быть ЛАТ ЗАГЛ, тогда аргумент функции, равный 1, дает переключение в режим РУС ЗАГЛ; -1- в режим РУС СТР; 0- в исходный режим ЛАТ ЗАГЛ. Чтобы не происходило путаницы, переход в режимы РУС ЗАГЛ и РУС СТР лучше делать из исходного режима (не сразу из РУС СТР в РУС ЗАГЛ или наоборот, а сначала выполнив USR0). Эта программа-один из немногих, видимо, способов управлять режимом работы клавиатуры непосредственно из программы на БЕЙСИКе или в кодах, так как БЕЙСИК не позволяет записывать числа непосредственно в старший байт ячейки ОЗУ. Думаю, что эта программа будет очень полезна для Бейсик-программ, связанных с вводом и обработкой текстов или просто при вводе ответа на запрос машины типа "Да" или "Нет" При этом

010737	002100	062737	000012	002100	000000	105703	002026
010046	010146	011501	005701	001407	002403	012701	000240
000404	012701	000200	000401	005001	110137	000043	005000
005037	000222	004737	110572	012601	012600	000207	000000

уменьшается вероятность ошибки пользователя-новичка при работе с этой программой, упрощается сама программа (становятся ненужными различные "ухищрения" для выявления возможных ошибок такого типа и "многобуквенные" инструкции с AND или OR в операторах для отработки и русских и латинских букв в ответе пользователя) и не надо "возиться" с переключением регистров.

2. При написании программ в кодах для выяснения режима работы клавиатуры и дисплея БК (РУС/ЛАТ, инверсия экрана, инверсия или подчеркивание символов и т.п.) обычно пользуются функцией EMT34, которая записывает в R0 слово состояния дисплея БК. После этого производя тестирование того или иного бита R0, можно определить включен или выключен тот или иной режим. (Кстати, выяснять значение первого и последнего бита проще и короче не обычной командой BIT #<маска>, R0 с последующим BEQ или BNE, а парой команд: (ROR или ROL) + (BCS или BCC)). Еще одно неудобство такого метода (кроме "громоздкости" и потери машинного времени на выполнение функции EMT34) в том, что узнать режим работы БК из программы на БЕЙСИКе можно, но для этого следует написать USR-программу. Более простой и быстрый способ, доступный и непосредственно в Бейсик-программе оператором PEEK или OUT, — обращение к байтам-признакам режимов в ОЗУ БК. Кстати, именно к этим байтам обращается EMT34, формируя слово состояния в R0. Далее приводится таблица с указанием адресов байтов-признаков, а также их значений для включенного режима.

Все адреса байтов — восьмеричные. При выключенном значении режима во всех случаях значение байта-признака равно 0. Значение при включенном режиме всюду -1(377_8), кроме номера 43, где -200_8.

3. Интересна ячейка ОЗУ @#162_8: записанная в ней константа зависит от режима "32/64 символа в строке" и определяет, на сколько байтов экранной памяти произвести сдвиг очередного знакоместа (курсора) при печати символов (ширина символа в режиме 64 символа в строке равна 1 байту, а в режиме 32 символа в строке — 2 байта). При изменении содержимого этой ячейки операторами MOVB в Ассемблере (кодах) и POKE в БЕЙСИКе можно при выводе текста получить интересные эффекты. В обычном состоянии значение байта @#162_8 равно 1 для 64 символов в строке и 2 — для 32 символов. Если она будет равна 2 или 4 (соответственно), то текст будет выведен на экран как бы вразрядку. Обычно этого добиваются, просто задавая текст в операторе PRINT уже с пробелами. Работая с ячейкой @#162_8, можно вывести текст в таком виде с экономией памяти вдвое. Другие значения этой константы приводят к тому, что либо печатаются два символа в одно место, либо они группируются в разделенные пробелом пары, либо просто при печати вразрядку расстояние между символами становится больше — это каж-

Таблица 1

Номер (адрес) байта-признака	Режим
40	32 символа в строке
41	Инверсия фона
42	Режим РП
43	РУС
44	Подчеркивание символов
45	Инверсия символов
46	ИНД СУ
47	БЛОК РЕД
50	ГРАФ
51	ЗАП
52	СТИР
53	32 символа в служебной строке
54	Подчеркивание в служебной строке
55	Инверсия символов в служебной строке
56	Гашение курсора

дый из читателей сможет выяснить сам, проведя простой эксперимент: POKE &O162, <конст.>. Установленный таким образом режим вывода действует всегда и в кодах, и в БЕЙСИКе, причем в БЕЙСИКе- и в операторе PRINT, и в INPUT, и при листинге программы, и в любом другом способе. Те из читателей, у кого есть принтер, могут поэкспериментировать: действует ли этот режим на вывод текста на принтер.

4. Фон экрана в БК-0010 при выполнении операций очистки экрана и печати символов определяется содержимым ячейки памяти @#212_8. При выполнении операции очистки экрана число из этой ячейки копируется во все ячейки ОЗУ экрана. При печати символа БК, готовя место для него, переписывает содержимое этой ячейки в ячейки ОЗУ экрана, соответствующие прямоугольнику в 1 символ. Затем в уже подготовленном знакоместе высвечиваются "текущим" цветом точки, соответствующие графической прорисовке данного символа.

Как легко заметить, получить при выводе текста в БЕЙСИКе на экране три цвета (например, весь экран синий, а символы красные на зеленом фоне) невозможно: оператор COLOR, как и "инв.э.", который используется оператором COLOR, меняет цвет фона на всем экране. Правда, есть способ вынести символы на экран цветом, отличным от цвета всего экрана и как бы в "окошке" черного цвета: перед печатью такого текста нужно дважды вывести код 155_{10} (или дважды нажав "АР2 + ;"), но после этого вернуться к обычному способу печати невозможно. Для примера: попробуйте поэкспериментировать с этой программкой на Бейсике с разными значениями C1 и C2, обозначающими номер цвета; программу просчитайте после вывода текста "СТОП в строке..." пошаговым методом с помощью клавиши ШАГ:

```
5 STOP
10 COLOR C1,C2
20 CLS
30 ? "AAA"
32 COLOR C1, 4 ' черный фон
35 COLOR C1, C2
40 ? CHR$(155); CHR$(155);
50 ? "БББ"
60 CLS
70 ? "CCC"
80 END
```

С помощью же ячейки 212_8 можно добиться любых эффектов такого рода: получать как общий фон экрана, так и фон знакоместа под символы любых цветов и расцветок хоть даже в вертикальную полоску! Этого мало! Экспериментируя, можно получить и другие эффекты, например эффект "утолщения" символов. Сделать это можно, используя оператор POKE или, что надежнее для БЕЙСИКа, подпрограмму в кодах. Вот текст программы для смены цвета фона:

```
120327 377 1012 10546 11505 42705 177760
6305 62705 37734 11567 140262 12605 207 0
177777 125252 52525 167356 156735 63146
154330 146314 104210 42104 0
```

Начальный адрес: 37700_8. После ввода с м/ф введите DEF USR = &O37700. Чтобы сменить цвет фона, вызовите функцию USR оператором типа A% = USR(A%), где A%-число целого типа, обозначающее цвет (см. таблицу 2). После этого все символы будут печататься в "окошке" заданного цвета. Если же вы после этого выполните команду CLS, то весь экран окрасится в заданный цвет, и если захотите, вы можете опять вызвать функцию USR и задать для знакоместа символа другой цвет.

Применять цвета 4-10 нужно осторожно: иногда символы на таком фоне видны плохо. Лучше использовать такие цвета для окраски всего экрана, а для знакомест символов выбирать все-таки обычные для БК цвета (а с другой стороны, при печати символов на фоне 4-10 можно получить некоторые интересные эффекты).

Для желающих поэкспериментировать (а также увидеть на экране обещанный фон "в разноцветную по-

Таблица 2

Значение A%	Цвет фона
0	Черный
1	Красный
2	Зеленый
3	Синий
4	Желтый
5	Фиолетовый
6	Голубой
7	Белый
8	Темно-красный
9	Темно-зеленый
10	Темно-синий
11 и более	Черный

лосочку)" привожу вторую программу:

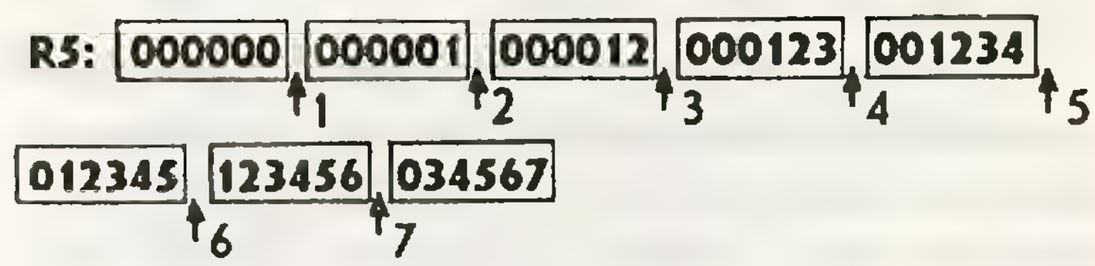
120327 000377 001005 010546 011505 010537 000212 012605 000207 000000

Начальный адрес любой; аргумент USR-функции — числа типов: %,&O или &B. Задавая разные значения аргумента, можно получать на экране все, что угодно. Введите в программу на БЕЙСИКе (пример, рассмотренный выше) дополнительную строку 25 B% = USR(B%) и попробуйте поэкспериментировать, подставляя различные значения B%.

5. В составе программного обеспечения БК, записанного в ПЗУ, есть полезные подпрограммы, которые можно вызывать из программ пользователя в кодах с помощью оператора JSR R7, @# <адрес>. Приведу данные о четырех из них:

а) Подпрограмма @#100460 — печать строки символов. При этом длина строки текста должна быть записана в первом байте строки, а затем должен идти сам текст (как для программирования ключей). Перед вызовом в регистр R3 нужно поместить адрес байта с длиной строки.

б) Подпрограмма @#100472 — ввод с клавиатуры восьмеричного числа. В обычных условиях ее использует монитор для ввода адреса запуска программы командой "C". После возврата введенное значение записано в R5. Если вводится "8", "9" или нецифровой символ, то в R5 тут же записывается 0 (Внимание! На экране внешне все нормально!) и вводить число придется заново. Если введено более шести цифр, то ввод каждой последующей цифры приводит к потере первой, второй и т.д. введенной цифры, а от цифры, записанной в самое левое "место" регистра, остается только ее младший бит:



(стрелкой показан ввод стоящей рядом с ней цифры). На последнем шаге приведенной диаграммы видно, что при вводе седьмой по счету цифры 7 цифра 1 потеряна, а от цифры 2 остался только ее младший бит: 0 ($2_{10} = 10_2$) Завершается ввод числа клавишей <ВВОД>

в) Подпрограммы @#110346 и @#110362 (вызов оператором JSR R4, @# <адрес>) сохраняют в стеке и извлекают из стека (соответственно) регистры R0-R3. При этом значение R4 не сохраняется — он используется как вспомогательная ячейка памяти. (Первая из подпрограмм — сохранение; вторая — восстановление).

И в заключение небольшой совет тем, кто тоже будет исследовать монитор и БЕЙСИК: лучше всего воспользоваться программой PATCH, входящей в прилагаемый к БК-0010.01 пакет программ на БЕЙСИКе. Нужно переписать коды и начальный адрес на бумагу (если есть принтер-великолепно! Поменяйте в программе команды PRINT на LPRINT и включите принтер). Затем коды нужно перевести в команды Ассемблера по таблицам, имеющимся в одной из брошюр серии "Микропроцессоры" (Издательство "Высшая школа"), посвященной микропроцессору К1801, и в книге Осетинского Л.Г. и др."Фокал для микро- и мини- компьютеров", о которой на страницах журнала говорилось уже неоднократно.

Нам пишут

В.В.Попов
(г.Кемерово)

БЫСТРЫЙ ВВОД ОПЕРАТОРОВ

В БЕЙСИКе БК-0010.01 для набора десяти наиболее часто употребляемых команд предусмотрены специальные функциональные клавиши (F0 — F9), обеспечивающие быстрый ввод этих команд. При необходимости функциональные клавиши можно перепрограммировать по своему усмотрению. Однако при написании программ на БЕЙСИКе десяти команд быстрого ввода явно недостаточно и остальные команды и операторы приходится вводить с клавиатуры посимвольно.

Предлагаемая подпрограмма — драйвер позволяет значительно расширить набор операторов БЕЙСИКа для быстрого ввода. Подпрограмма — драйвер работает совместно с БЕЙСИКом "Вильнюс, 1986607.24", зашитым в ПЗУ компьютера, так как в подпрограмме есть обращения к ПЗУ БЕЙСИКа.

```
10 DATA 5599,266,48,119,188,2335,-
28448,4454,2527,-31718,-24553,3,525,-
30714,5570,26,4227,26051,64,-24573,775,26051,
32,-24573
20 DATA 771,32394,95,-
32152,24706,7351,400,80,5599,336,24,-24640,70,-
30025,66,498,5599,-
32694,24,5047,22,2807,18,2807, 14,8183
30 DATA 10,8,9719,-
30714,2,773,0,5599,336,24,2,-24640,18,-30025,14,-
23104,32,757,5599,-32694,24,2,0,-12664,-12624,-
18758,-18704
40 DATA -14630,-18951,-14646,-12616,-
18933,-18829,-18806, -18824,-12643,-18910,-
18923,-18899,-18736,-18884,-18856, -14635,-
14587
50 DATA -12660,-18710,-18961,-12649,-
18977,5569,468,5570,14, -30704,95,-24442,-
28404,-27821,-28789,29285,19345,26003,
28295
60 FOR A% = 256 TO 480 STER 2
70 READ D%
80 PORE A%,D%
90 NEXT A%
```

После запуска программы в стековой области формируется подпрограмма в машинных кодах. Ее необходимо записать на магнитофон командой:

BSAVE "SKEY:,256,482

Для загрузки драйвера с магнитофона наберите команду:

BLOAD "SKEY",R

Теперь для быстрого ввода оператора необходимо нажать "префикс" (клавиша "КТ"), а затем одну из 26 букв латинского алфавита.

Соответствие букв и "приклееных" к ним операторов приведено в таблице 1. Там же приведены адреса букв в драйвере.

Таблица 1

A,402- ASC	H,416- HEX	O,430- OPEN
V,444- VAL		
B,404- BIN	I,418- INPUT	P,432- PRINT
W,446- RENUM		
C,406- CSAVE	J,420- DATA	Q,434- NEW
X,448- DEF		
D,408- DELETE	K,422- KEY	R,436- RETURN
Y,450- INREY		
E,410- ELSE	L,424- LOCATE	S,438- STOP
Z,452- DIM		
F,412- FOR	M,426- MID	T,440- THEN
G,414- GOSUB	N,428- NEXT	U,442- USR

В качестве примера вызовем команду RENUM: нажимаем и отпускаем клавишу "КТ", а затем нажимаем клавишу "W" (или "w") — на экране появляется полное слово команды, если теперь нажать клавишу "ВВОД", то выполнится команда перенумирования строк. Другой пример: необходимо ввести строку программы вида:

10 IF I = 1 THEN 20 ELSE PRINT A.

Для этого необходимо выполнить следующую последовательность:

10 IF I = 1 "КТ" T 20 "КТ" E "КТ" P A.

Для тех читателей, которых не удовлетворят "приклеенные" операторы, автор советует выбрать из предлагаемых 108 операторов, функций и команд языка БЕЙСИК необходимые, "приклеить" их к выбранной клавише и записать полученную конфигурацию подпрограммы — драйвера на магнитофон. Так можно создать несколько версий драйвера и загружать нужную при работе в БЕЙСИКе.

Для этого необходимо: оператором БЕЙСИКа POKE по адресу буквы (взятому из табл.1) записать адрес начала оператора (из табл.2). Например "приклеем" к букве O оператор RESTORE:

POKE 430,-14654

Таблица 2

ABS.....-12715	DATA....-18829	LET.....-18927
PRESET..-18782		
AND.....-12777	DEF.....-18961	LINE....-18973
PRINT...-18899		
AS.....-14590	DELETE..-18704	LIST....-18727
PSET....-18787		
ASC.....-12664	DIM.....-18977	LOAD....-18746
READ....-18870		
AT.....-14618	DRAW....-18893	LOCATE..-18824
REM.....-14670		
ATN.....-14622	EL.....-14625	LOG.....-12727

RENUM...-18710
AUTO....-18732 ELSE....-14630 LLIST...-18716
RESTORE.-14654
BEEP....-18802 END.....-18833 LPOS....-12677
RETURN..-14666
BF.....-14578 EOF.....-12692 LPRINT..-18851
RND.....-12755
BIN.....-12624 EQV.....-12788 MID.....-12643
RUN.....-18692
BLOAD...-18769 EXP.....-12731 MONIT...-18722
SAVE....-18741
BSAVE...-18752 FIND....-18763 MOD.....-12800
SGN.....-12711
CALL....-18875 FIX.....-12719 NEXT....-14659
SIN.....-12747
CDBL....-12702 FN.....-14583 NEW.....-18736
SQR.....-12751
CYR.....-12628 FOR.....-18951 NOT.....-12796
STEP....-14608
CINT....-12707 FRE.....-12672 OCT.....-12620
STOP....-18856
CIRCLE..-18968 GOSUB...-14646 ON.....-18844
STR.....-12632
CLEAR...-18817 GOTO....-14640 OPEN....-18923
STRING..-12639
CLOAD...-18775 HEX.....-12616 OR.....-12780
TAB.....-14615
CLOSE...-18916 IF.....-18936 OUT.....-18860
TAN.....-12739
CLS.....-18797 INKEY...-12649 OUTRUT..-14603
THEN....-14635
COLOR...-18793 IMP.....-12792 PAINT...-18907
TO.....-14611
CONT....-18697 INP.....-12764 PEEK....-12760
TROFF...-18957
COS.....-12743 INPUT...-14596 PI.....-12773
TRON....-18811
CSAVE...-18758 INT.....-12723 POINT...-12770
ISR.....-14587
CSNG....-12697 KEY.....-18806 POS.....-12681
VAL.....-12660
CSRLIN..-12688 LEN.....-12668 POKE....-18865
XOR.....-12784

В качестве префиксной клавиши можно выбрать любую другую клавишу, записав по адресу — 278 код этой клавиши. Определим в качестве префиксной — клавишу "ВС" (десятичный код — 19): POKE 278,19

В заключение следует отметить, что выключается драйвер командой POKE 48,-32162

Заключение рецензента

Набор программы (особенно большой) на клавиатуре БК0010 — достаточно утомительное занятие. Программа-драйвер, предлагаемая автором, позволяет несколько облегчить этот нелегкий труд. Большой выгоды от такого новшества нет, но клавиатура БК работает в более щадящем режиме. Экономится время ввода команд.

Л.Н.Жариков

А.В.Ильин
(г.Воркута)

Предлагаю в программах для БК-0010.01 перед каждым запуском на выполнение проверять содержимое ячейки &040, в которой записана информация о текущем состоянии режима 32/64. При этом появляется возможность автоматического переключения режимов независимо от текущего режима. Это повысит уровень сервиса программ.

Например, для программ на языке BASIC, работающих в режиме 32 символа в строке, эту операцию можно реализовать так:

```
1 IF PEEK (&040) = 0 THEN ? CHR$(&0233)
10 <ТЕКСТ ПРОГРАММЫ>
.....
```

То же для программы на языке ФОКАЛ:

```
1.1 I (FX(1,40)),1.2,10.10
1.2 X FCHR(155)
10.10 <ТЕКСТ ПРОГРАММЫ>
.....
```

А.В.Алексеев
(г.Ленинград)

О ТЕКСТАХ ПОДПРОГРАММ В КОДАХ

Меня заинтересовала статья А.П.Гармашова "О текстах подпрограмм в кодах для БЕЙСИКа БК 0010.01". Однако, как подметил автор статьи, программа имеет ряд недостатков: длина исходной подпрограммы в машинных кодах должна быть не более &O212, а также программа генерации теряется. Все эти недостатки можно устранить, применив другой принцип для построения программы, как мне кажется, более простой и очевидный. Применив этот принцип, я разработал две программы, отличающиеся друг от друга лишь степенью сервиса (вторая программа имеет более удобный интерфейс к пользователю). Из программы автора статьи я позаимствовал лишь подпрограмму USR2 — вызова процедуры монитора чтения файла с магнитной ленты и идею буферизации подпрограммы в кодах в экранной области ОЗУ.

Теперь более подробно о принципе, который положен в основу моей программы. Как известно из руководства по языку БЕЙСИК (1986.07.24), файлы данных записываются на магнитную ленту в формате кодов ASCII, также как и текстовые файлы программ, записанные командой SAVE. Из этого можно сделать следующий вывод: Бейсик-программы можно создавать средствами символьных функций Бейсик-программы, накапливая строки создаваемой программы в символь-

ном массиве (каждый элемент массива — строка создаваемой программы). Затем можно с помощью операторов работы с файлами данных записать этот массив на магнитную ленту. Соответствующий фрагмент программы будет выглядеть следующим образом:

```
OPEN R$ + '.ASC' FOR OUTPUT
FOR I% = 1% TO V%
? # A$(I%)
NEXT I%
CLOSE
```

где R$ — символьная переменная, содержащая имя файла; V% — размер символьного массива; A$(N) — символьный массив, содержащий строки создаваемой программы.

Далее записанную программы можно считать с магнитной ленты командой LOAD, дополнить ее необходимыми операторами и скомпилировать подпрограмму в кодах, как в программе автора статьи.

Мною были написаны две программы: одна максимально минимизированная и позволяющая генерировать большие программы, другая — предоставляющая пользователю ряд удобств в работе с ней и занимающая больше места в памяти, хотя я предпринял все меры по сокращению памяти, занимаемой программой генерации.

Теперь рассмотрим более подробно каждую из программ. Разбор начнем с первой, короткой программы. Операторы в строках 10 и 20 устанавливают размер памяти, выделяемой под хранение строк генерируемой программы. В строках 30 — 70 загружается и определяется программа в машинных кодах, вызывающая процедуру чтения файла с магнитной ленты. Затем производится очистка экранной памяти, загрузка упаковываемой подпрограммы в машинных кодах в экранное ОЗУ с адреса &O42000. Далее в строках 130 — 230 происходит формирование очередной строки генерируемой программы. Причем каждая строка содержит: номер, оператор DATA и восемь чисел. Оператор в строке 240 удаляет последнюю запятую в последней неполной строке. Если строка полная, то после последнего числа нет запятой (см.строку 200). Потом полученный символьный массив записывается на магнитную ленту в файле с расширением '.ASC'. В конце программа запрашивает пользователя о дальнейших действиях. Вторая программа содержит первую в качестве одной из подпрограмм. Выбор требуемой функции осуществляется с помощью меню. Курсор перемещается вверх и вниз с помощью двух клавиш 'курсор вверх' и 'курсор вниз' и устанавливается на необходимую строку. По нажатии клавиши 'ВВОД'(ВК) осуществляется переход на необходимую подпрограмму. С помощью второй программы можно кроме упаковки подпрограмм в кодах распечатать сгенерированную программу на экран или принтер записать программу на магнитную ленту.

Обе программы были мною проверены на подпрограмме, реализующей команду MERGE (ВТ и ее применение. — № 4. — 1990.), которой я пользовался до создания моих программ. Я решил добиться аналогичного результата с помощью программы генерации. Последовательность действий при проверке была такая:

1. С помощью отладочного монитора я записал в память подпрограмму с адреса &01000 и директивой М3 вывел ее на магнитную ленту.

2. Отключив блок МСТД, запустил БЕЙСИК и загрузил свою программу генерации.

3. Загрузил подпрограмму в кодах, которая моей программой была упакована в операторы DATA.

4. Вывел сгенерированную программу на магнитную ленту.

5. Считал сгенерированную программу командой LOAD, дополнил ее необходимыми операторами и скомпилировал.

Подпрограмма оказалась полностью работоспособной. Я думаю, что эти две программы генерации окажутся полезными пользователям БК, использующим подпрограммы в машинных кодах.

Программа 1

```
10 CLEAR 2000                          'Выделение памяти под массив
20 DIM A$(30)                          'Объявление массива
30 POKE &O400,&O11505                  'Загрузка
40 POKE &O402,&O4737                   'подпрограммы
50 POKE &O404,&O100536                 'вызова п/п
60 POKE &O406,&O207                    'загрузки файла
70 DEF USR = &O400                     'ее определение
80 ? CHR$(140) + CHR$(140);            'Очистка ОЗУ экрана
90 A% = USR(&O42000)                   'Вызов п/п загрузки
100 DL% = &O42000 + PEEK(&O266)        'Нач.адр. считывания + длина файла
110 C% = &O42000                       'Адрес первого упаковываемого маш. слова
120 I% = 10%                           'Номер первой строки генерир.программы
```

```
130 V% = I%\10%                                  'Индекс элемента символьного массива
140 A$(V%) = " "                                 'Очистка очередного элемента массива
150 A$(V%) = " " + MID$(STR$(I%),2%,LEN(STR$(I%))) + "DATA"
160 FOR J% = 1% TO 8%                            'Цикл формирования очередной строки
170 A$(V%) = " "A$(V%) + "&O" + OCT$(PEEK(C%))   'генерируемой
180 IF J% < 8% THEN A$(V%) = " " + A$(V%) + ","  'программы
190 C% = C% + &O2                                'Адрес следующего машинного слова
200 IF C% > DL% THEN 240                         'Если все слова прочитаны то на выход
210 NEXT J%
220 I% = I% + 10%"                               'Номер следующей строки
230 GOTO 130                                     'Идти на формирование новой строки
240 IF J% < 8 THEN A$(V%) = " " + MID$(A$(V%),1%,LEN(A$(V%))-1)
250 CLS
260 R$ = " "
270 INPUT "ИМЯ ФАЙЛА"; R$
280 ? "ВКЛЮЧИ МАГНИТОФОН."
290 IF INKEY$ = " " THEN 290
300 OPEN R$ + ".ASC" FOR OUTPUT                  'Запись
310 FOR J% = 1 TO V%                             'Массива
320 PRINT # A$(J%)                               'на
330 NEXT J%                                      'магнитную ленту
340 CLOSE
350 INPUT "БУДЕТЕ ПРОДОЛЖАТЬ";U$                 'Запрос о дальнейших действиях
360 IF U$ = "D" OR U$ = "Д" THEN 80
370 CLS
380 END
```

Программа 2

```
10 REM ПРОГРАММА УПАКОВКИ П/П В КОДАХ В ОПЕРАТОРЫ 'DATA'
20 REM АЛЕКСЕЕВ А.В.1.10.1990.              БЕЙСИК-БК
30 CLEAR 2000
40 DIM A$(30)
50 POKE &O400,&O11505
60 POKE &O402,&O4737
70 POKE &O404,&O100536
80 POKE &O406,&O207
90 DEF USR = &O400
100 CLS
110 ? AT(0,6)" ЗАГРУЗКА И УПАКОВКА П/П < = / *"        'Печать меню
120 ? AT(0,8)" РАСПЕЧАТКА НА ЭКРАН     < = / *"
130 ? AT(0,10)" РАСПЕЧАТКА НА ПРИНТЕР    < = / *"
140 ? AT(0,12)" ЗАПИСЬ НА МАГНИТОФОН     < = / *"
150 ? AT(0,14)" ВЫХОД                  < = / *"
160 LINE (0,50)-(245,157),3,B                          'Рамка меню
170 Y% = 6%
180 LOCATE 29%,Y%,1%                                   'Высвечивание курсора
190 T$ = INKEY$                                        'Считывание символа с клавиатуры
200 IF T$ = " " THEN 190
210 T% = ASC(T$)                                       'Определение кода символа
220 LOCATE 29%,Y%,0%                                   'Гашение курсора
230 IF T% = 26% THEN 270                               'Переход на
240 IF T% = 27% THEN 310                               'соответствующую программу в
250 IF T% = 10% THEN 350                               'зависимости от нажатой клавиши
260 GOTO 180
270 Y% = Y%-2%                                         'П/п движения курсора вверх
280 IF Y% = 4% THEN BEEP ELSE 300
290 Y% = 6%
300 GOTO 180
310 Y% = Y% + 2%                                       'П/п движения курсора вниз
320 IF Y% = 16% THEN BEEP TLSE 340
330 Y% = 14%
340 GOTO 180
350 ON Y%\2%-2% GOTO 360,660,750,800,920               'П/п перехода на требуемую функцию
```

```
360 CLS                                   'П/п упаковки п/п в машинных кодах
370 ? AT(8,0)"ФОРМАТ УПАКОВКИ"
380 ? AT(13,2)"1-OCT"
390 ? AT(13,3)"2-HEX"
400 ? AT(13,4)"3-DEC"
410 B$ = INKEY$
420 IF B$ = " "THEN 410
430 IF B$<"1" OR B$>"3" THEN 360
440 ? CHR$(140) + CHR$(140);
450 LOCATE 0%,10%
460 A% = USR(&O42000)
470 DL% = &O42000 + PEEK(&O266)
480 C% = &O42000
490 I% = 10%
500 V% = I%\10%
510 A$(V%) = " "
520 A$(V%) = " " + MID$(STR$(I%),2%,LEN(STR$(I%))) + "DATA "
530 FOR J% = 1% TO 8%
540 D% = PEEK(C%)
550 IF B$ = "1" THEN A$(V%) = " " + A$(V%) + "&O" + OST$(D%)
560 IF B$ = "2" THEN A$(V%) = " " + A$(V%) + "&H" + HEX$(D%)
570 IF B$ = "3" THEN IF D%<0%
    THEN A$(V%) = " " + A$(V%) + MID$(STR$(D%),1%,LEN(STR$(D%))-1) + "%"
    ELSE A$(V%) = " " + A$(V%) + MID$(STR$(D%),2%,LEN(STR$(D%))-2) + "%"
580 IF J%<8% THEN A$(V%) = " " + A$(V%) + ","
590 C% = C% + &O2
600 IF C%>DL% THEN 640
610 NEXT J%
620 I% = I% + 10%
630 GOTO 500
640 IF J%<8% THEN A$(V%) = " " + MID$(A$(V%),1%,LEN(A$(V%))-1)
650 GOTO 100
660 CLS                                   ' П/п печати сгенерированной программы на экран
670 FOR J% = 1% TO V%
680 ? A$(J%)
690 IF CSRLIN<22 THEN 720
700 IF INKEY$ = " " THEN 690
710 CLS
720 NEXT J%
730 IF INKEY$ = " " THEN 730
740 GOTO 100
750 CLS                                   'П/п печати сгенерированной программы на принтер
760 FOR J% = 1% TO V%
770 LPRINT A$(J%)
780 NEXT J%
790 GOTO 100
800 CLS                                   'Запись сгенерированной программы на магнитную ленту
810 R$ = " "
820 LOCATE 0%,10%
830 INPUT "ВВЕДИТЕ ИМЯ ФАЙЛА";R$
840 ? "ВКЛЮЧИТЕ МАГНИТОФОН И НАЖМИТЕ ЛЕВУЮ КЛАВИШУ"
850 IF INKEY$ = " " THEN 850
860 OPEN R$ + ".ASC" FOR OUTPUT
870 FOR J% = 1% TO V%
880 PRINT # A$(J%)
890 NEXT J%
900 CLOSE
910 GOTO 100
920 CLS                                   'Очистка экрана
930 END                                   'и выход из программы
```

ВЫПУСК 1: По страницам журнала КОМПЬЮТЕРНЫЙ МОСТ ПОД РЕДАКЦИЕЙ Г.Р.ГРОМОВА

К выходу в свет первого БИТа

На прилавках книжных магазинов появилась первая книга из новой серии БИТ ("Библиотека Информационной Технологии" — БИТ/1 /Под ред. Г. Р. Громова. — М.: Наука, ФИЗМАТЛИТ, 1990. — 208 с., ил., 100 000 экз., 4 руб. 20 коп.).

О чем эта книга? Если коротко — о персональных компьютерах. Вот несколько примеров.

Книга открывается размышлениями известного советского программиста А.Чижова об особенностях работы системы MS DOS. Далее А.Чижов знакомит читателей с этапами развития MS DOS, дает практические советы программистам, работающим с IBM PC...

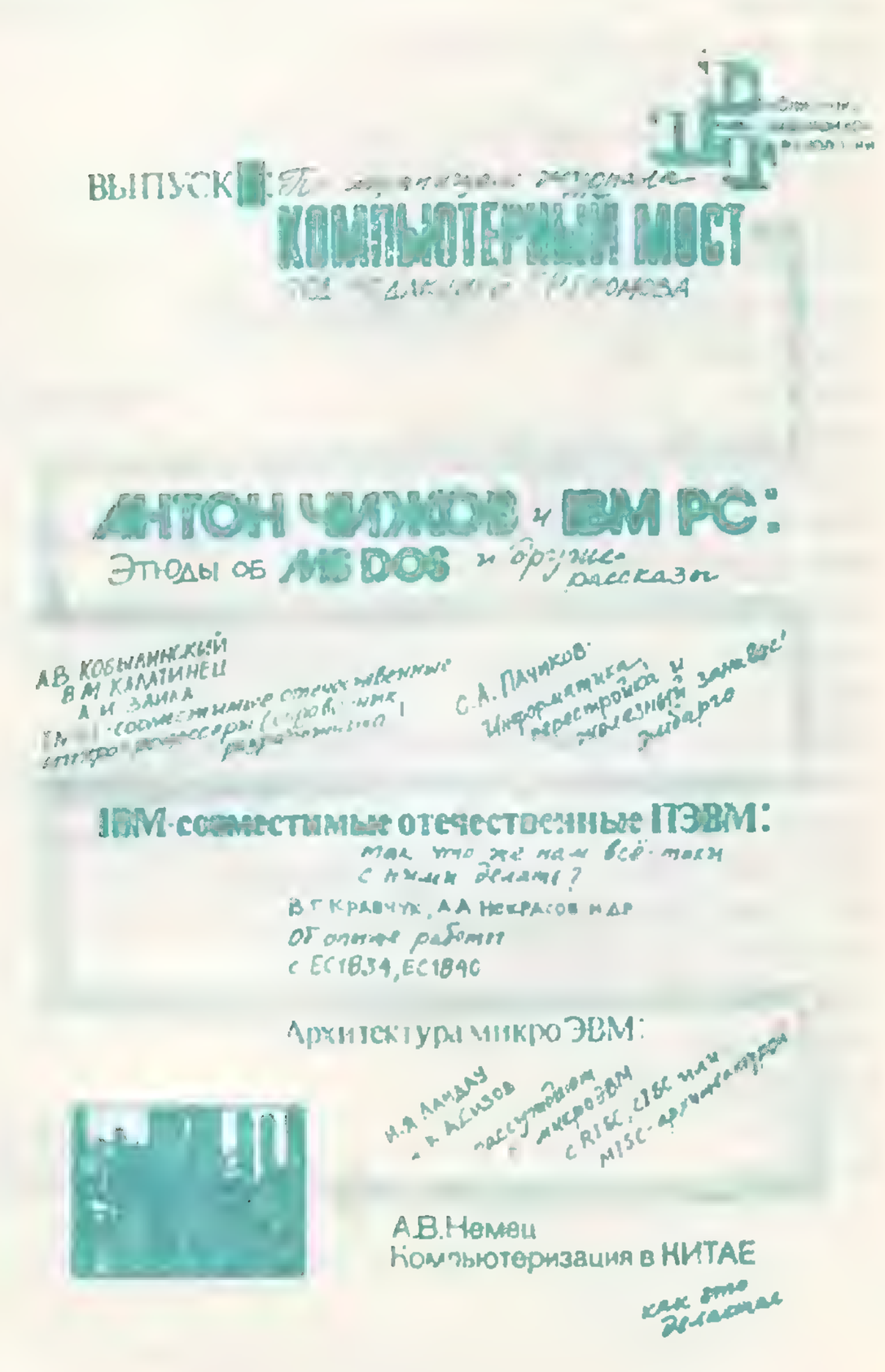
Э.М.Пройдаков анализирует первые три года развития PS/2 — второго поколения ПЭВМ фирмы IBM.

А.В.Кобылинский и другие киевские специалисты работают над рядом отечественных микропроцессоров, программно совместимых с Intel 8086, 8087 и т.д. (на базе которых создана значительная часть мирового парка компьютеров). В первом выпуске БИТ описаны особенности архитектуры, система команд, временные диаграммы микропроцессора КМ1810ВМ87.

В БИТе есть статьи об опыте работы с ПЭВМ "Роботрон" 1715 и IBM-совместимыми компьютерами ЕС1834, ЕС1840, полемика И.Я.Ландау и К.А.Сизова о микроЭВМ с RISC, CISC или MISC-архитектурой, реплика С.А.Пачикова об информатике и эмбарго КОКОМ, две статьи А.В.Немеца о социально-экономических проблемах информатизации в КНР...

Книга полезна и специалистам в области ПЭВМ, и широкому кругу пользователей, только еще начинающих осваивать персональные компьютеры. Это первый выпуск БИТа, и конечно, возникает вопрос: "А что дальше?". В следующих выпусках читатели познакомятся с новыми этюдами Чижова, техникой освоения систем подготовки текстов в MS WORD, особенностями программирования на Паскале, работой с национальными алфавитами на ПЭВМ, генераторами экспертных систем, организацией диалога...

Как приобрести БИТ/1? Чтобы получить по почте первый выпуск БИТ, надо перечислить по 5 руб. за каждый экземпляр на счет предприятия "РАТЕКС" (р/с 341116 в Краснопресненском отд. ЖСБ г.Москвы, МФО 20114, код 9201142) и выслать по адресу предприятия (127486, Москва, ул.Дегунинская, д.1/4, предприятие "РАТЕКС") заявку с точным адресом подписчика и копией платежного документа.



как «убить» МАШИННОЕ время

Е.В.Воронин
(г.Омск)

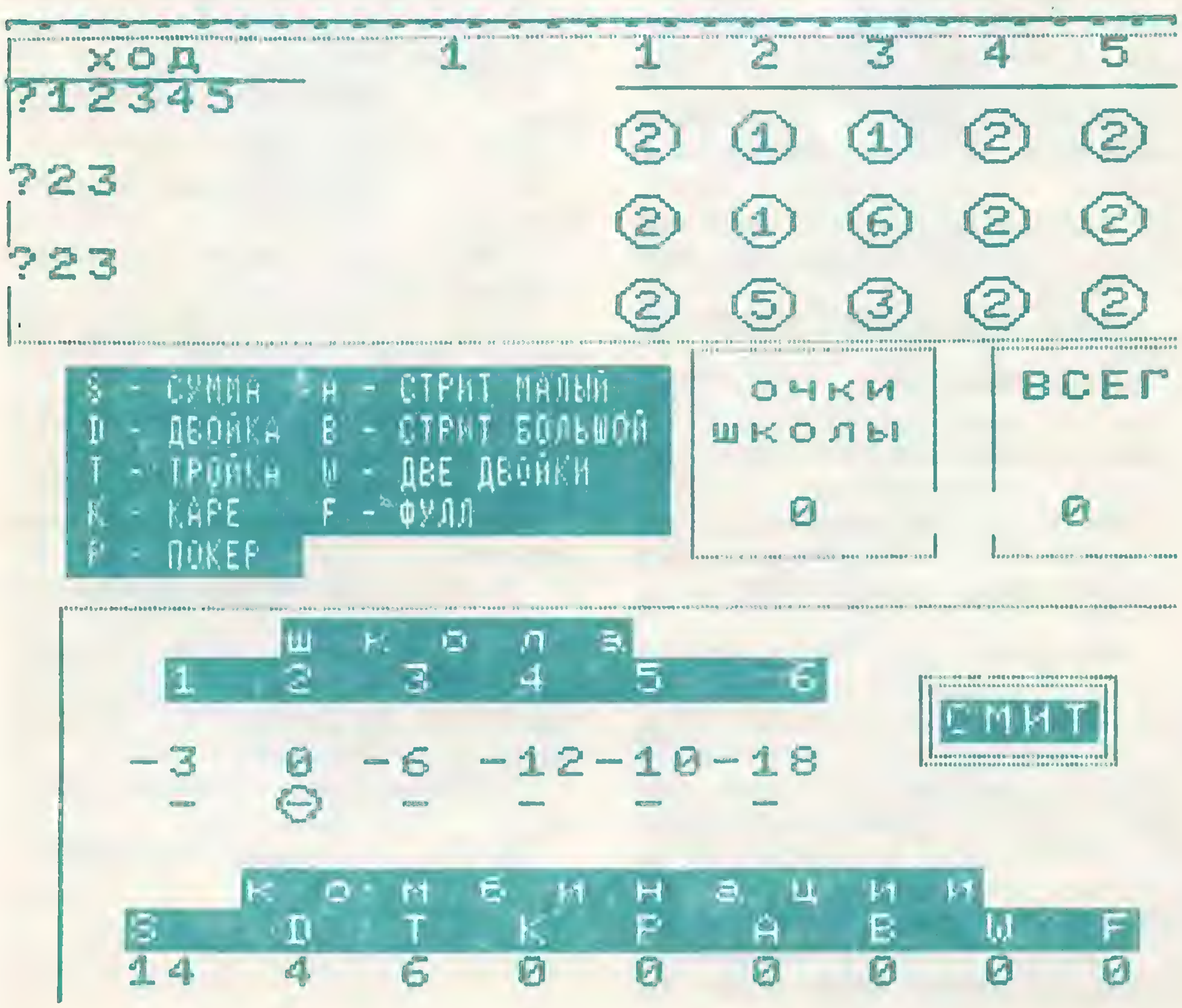
Игра в покер на БК

Представленная программа написана на языке БЕЙСИК для БК-0010.01. При ее написании использовалась статья А.Лашкова "Игра ПОКЕР". Поэтому пользователю для работы с ней вначале следует обратиться к журналу "Вычислительная техника и ее применение" (1990 г. № 7). А в конце этой статьи напечатано письмо А.Лашкова со списком опечаток, допущенных в тексте программы.

Игра проходит в диалоговом режиме. Перед игрой ЭВМ просит указать количество игроков, а также число людей, сидящих за компьютером. Недостающее число участвующих в игре компьютер промоделирует сам и будет за них вести всю игру. Это важное отличие от описанной у А.Лашкова программы. Затем ЭВМ предложит представиться игрокам, т.е. указать свое имя (не более семи символов).

Игра проходит в 15 кругов, при этом очередность хода в круге каждый раз меняется в зависимости от предыдущего результата.

На экране представлено текущее состояние игры: в левой верхней части ход, номер круга, в правой верхней части результаты броска, т.е. показания пяти костей. В нижней части экрана оперативно меняется информация о результате текущего броска,



т.е. указано, какая комбинация сколько очков имеет в данном случае. Там же будет обведено имя игрока, делающего ход. В середине экрана — справочная информация: названия комбинаций, общее число очков игрока и количество очков за школу.

Ход делается указанием номеров костей, которые нужно сбросить. Например, 245 "ВВОД" — бросить кости 2, 4, 5. Если ход удовлетворяет игрока и последующих попыток не будет, то он должен ввести нуль. После этого курсор исчезнет, а под комбинацией "школа 1" (внизу экрана) появится кружок. Внутри кружка может быть "+" или "-", означающие, выполнена или невыполнена данная комбинация. Кружок можно передвигать по нижней части экрана клавишами джойстика (со стрелками). Если кружок остановить под какой-либо комбинацией с "-" и нажать "ВВОД" или пробел, то данная комбинация будет зачтена, а количество очков за нее будет зачислено игроку. На 15-м кругу производится учет состояния очков за школу. Если школа невыполнена, то игрок штрафуется на 50 очков.

В конце круга появляется итоговая таблица, в которой сказано про каждого игрока:

сколько очков в данном круге набрано;

сколько очков всего;

сколько очков за школу;

очки премии за круг и общее число очков с учетом премии.

Премия за круг введена в программу с тем, чтобы обострить игру и подсчитывается по формуле:

$$PR_i = [2 \cdot O_i - O_{min} - O_{max}]/5,$$

где O_i — очки за игру, O_{min}, O_{max} — минимально и максимально набранное количество очков за игру. Таким образом, лидер в круге получает премию 20% от набранной им разницы с отставшим в круге. Премию получат и игроки, близкие по очкам к лидеру. Следует сказать, что в следующем круге лидер будет делать свою игру последним, что дает ему возможность проследить игру своих партнеров.

В конце игры ЭВМ определит и выдаст имя победителя.

Теперь об игре самой ЭВМ. Она осуществляется программированием алгоритма выбора действий за игрока, компьютером и фиксированными числами определяющими сам выбор действий. Так, если в операторе DATA (строка 2) изменить каждое второе число, начиная с первого, на меньшее, то можно "ослабить" игру ЭВМ. Правда, слишком большое увеличение этих чисел также нежелательно, так как имеют свои пределы, определяемые максимумом очков, которые можно получить за данную комбинацию. Так, за покер максимально можно получить 110 очков (6 6 6 6 6 с первой попытки). Значит, число 17.5 можно заменить числом от 0 до 110.

Небольшой комментарий по самой программе: строки 1-39 — инициализация игроков;

40-57 — построение "картинки" игры;

58-63 — ввод хода;

64-72 — "расшифровка" введенного хода;

73-80 — ход и вывод результатов в правую часть;

81-125 — обработка броска и вывод на экран результатов комбинаций;

126-148 — ввод и формирование реакции ЭВМ при выборе комбинации;

149-151 — подпрограмма "задержки";

152-166 — обработка выбранной комбинации;

167-174 — определение последовательности игроков в следующем круге, т.е. кто больше, кто меньше набрал за круг очков;

175-186 — вывод итоговой таблицы;

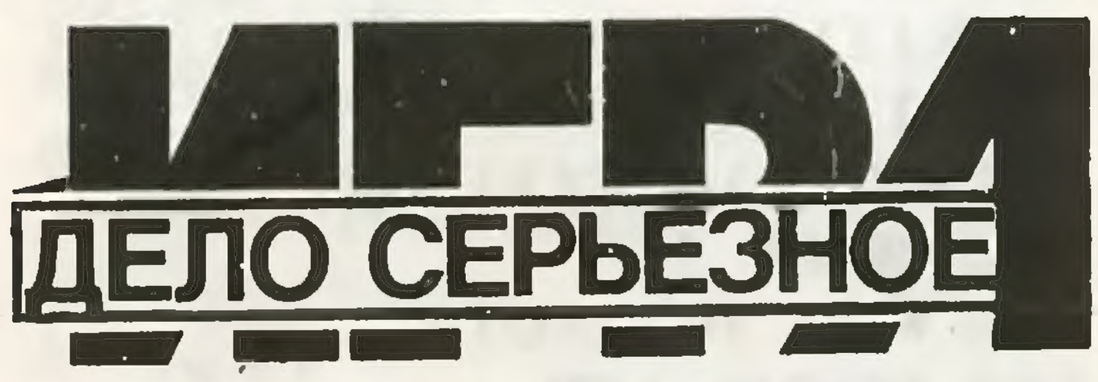
190-216 — алгоритм игры ЭВМ.

```
0 REM ПРОГРАММА  ИГРЫ  В
 ПОКЕР.   Автор Воронин Е
.В.      1990г.
1 DIM U%(15%),V%(15%,6%),
RE(15%),P%(15%),O%(15%)
2 DATA 20%,9%,14%,8%,21%,
6%,20%,5%,17.5,3%,15%,1%,
20%,2%,27%,7%,34%,4%
3 FOR I=1%TO6%
4 P%(I)=I+9%
5 O%(I)=7%-I
6 NEXT I
7 P%(0)=16%
8 QQ=6%
9 CLS
10 POKE 112%,15360%+PEEK(
132%)
11 ? "              - П О К Е
 Р -
12 ? AT(3,2)"Сколько игро
ков [1..6]";
13 J¤(1)="СМИТ"
14 J¤(2)="ДЮПОН"
15 J¤(3)="КИМ"
16 J¤(4)="ТЕЙЛОР"
17 J¤(5)="БЕРТОНИ"
18 J¤(6)="ВОЛЬФ"
19 FOR I=7%TO15%
20 READ RE(I),P%(I)
21 O%(16%-P%(I))=I
22 NEXT I
23 INPUT QQ
24 IF QQ<1%ORQQ>6%TH QQ=6
%
25 ? AT(3,3)"Сколько чело
век [0..6]";
26 INPUT NN
27 IF NN=>QQ TH32
28 FOR I=1%TOQQ-NN
29 ? AT(8%,4%+I)I;J¤(I)
30 NEXT I
31 IF NN<=0%TH36
32 FOR I=QQ-NN+1%TOQQ
33 ? AT(8%,I+4%)I;
34 INPUT J¤(I)
35 NEXT I
36 FOR I=1%TOQQ
37 PX(I)=I
38 NEXT I
39 GOSUB 149
40 FOR G%=1%TO15%
41 FOR KK=1%TOQQ
42 NK=PX(KK)
43 CLS
44 LINE (10%,137%)-(250%,
239%),2%,B
45 ? AT(7,14)CHR¤(156%);"
ш к о л а"AT(4,15)"1  2
 3  4  5   6"AT(3,21)"S
 D  T  K  P  A  B  W  F"AT
(24,16)J¤(NK)AT(6,20)"к о
 м б и н а ц и и
46 ? CHR¤(155%)AT(3%,8%)"
 S - СУММА  "AT(3%,9%)" D
 - ДВОЙКА "AT(3%,10%)" T
- ТРОЙКА "AT(3%,11%)" K -
 КАРЕ   "
47 ? AT(3%,12%)" P - ПОКЕ
P  "AT(15%,8%)" A - СТРИТ
 МАЛЫЙ   "AT(15%,9%)" B -
 СТРИТ БОЛЬШОЙ "
48 ? AT(15%,10%)" W - ДВЕ
 ДВОЙКИ    "AT(15%,11%)"
F - ФУЛЛ            ";CHR¤(
156%);CHR¤(155%)
49 ? AT(19%,8%)"очки   ВС
ЕГО"AT(18,9)"школы"AT(19
50 ? AT(0%,0%)"  ход
      1  2  3  4  5"AT(10
%,0%)G%AT(0%,1%)"?12345"A
T(0%,1%)
51 LINE (125%,12%)-(245%,
12%)
52 LINE (140%,75%)-(190%,
125%),2%,B
53 LINE (202%,75%)-(250%,
125%),2%,B
54 LINE (187%,154%)-(195%
+LEN(J¤(NK))*8%,175%),2%,
B
55 LINE (189%,156%)-(193%
+LEN(J¤(NK))*8%,173%),2%,
B
56 LINE (0%,0%)-(250%,73%
),2%,B
57 LINE (0%,10%)-(60%,10%
)
```

```
58 J%=1%
59 IF J%=1%TH135EL A¤=""
60 ? AT(0%,J%*2%-1%)"
   "AT(0%,J%*2%-1%);
61 IF NK<=QQ-NN TH190
62 INPUT A¤
63 IF A¤=""TH60
64 IF A¤="0"TH129
65 N%=LEN(A¤)
66 E=0%
67 FOR I=1%TON%
68 K%=ASC(MID¤(A¤,I,1%))
69 IF K%<48%ORK%>57%TH E=
1%
70 A%(I)=K%-48%
71 NEXT I
72 IF E=1%TH59
73 FOR I=1%TON%
74 B%(A%(I))=RND(1)*6%+1%
75 NEXT I
76 ? AT(15%,J%*2%);
77 FOR I=1%TO5%
78 ? B%(I);
79 CIRCLE (107%+I*24%,J%*
20%+4%),6%
80 NEXT I
81 ?
82 FOR I=1%TO15%
83 U%(I)=0%
84 NEXT I
85 FOR I=1%TO6%
86 C%(I)=0%
87 NEXT I
88 FOR I=1%TO5%
89 C%(B%(I))=C%(B%(I))+1%
90 U%(7%)=U%(7%)+B%(I)
91 NEXT I
92 Y%=0%
93 FOR I=1%TO6%
94 U%(I)=I*(C%(I)-3%)
95 ? AT(I*3%,17%)U%(I)AT(
I*3%+1%,18%);
96 IF V%(I,NK)=0%TH?"-"EL
?"+
97 IF C%(I)<2%TH IFC%(I)=
1%TH Y%=Y%+I EL Y%=Y% EL
Y%=-1%
98 NEXT I
99 ? AT(2%,22%)U%(7%)AT(4
%,23%);
100 IF V%(7%,NK)=0%TH?"-"
;EL?"+";
101 IF Y%=15%ORY%=20%TH U
%(Y%/5+9%)=Y%
102 IF Y%=15%ORY%=20%TH11
1
103 FOR I=1%TO6%
104 IF C%(I)=2%ANDU%(8%)>
0%TH U%(14%)=U%(8%)+I*2%
105 IF C%(I)=2%ANDI*2%>U%
(8%)TH U%(8%)=I*2%
106 IF C%(I)>2%TH U%(C%(I
)+6%)=C%(I)*I
107 NEXT I
108 IF U%(8%)=0%ORU%(9%)=
0%TH111
109 U%(15%)=U%(8%)+U%(9%)
110 U%(14%)=U%(8%)+U%(9%)
/3%*2%
111 FOR K=1%TO6%
112 IF C%(K)<3%TH116
113 FOR J=2%TOC%(K)-1%
114 IF U%(6%+J)<K*J TH U%
(6%+J)=K*J
115 NEXT J
116 NEXT K
117 IF U%(10%)>0%TH U%(14
%)=U%(10%)
118 IF U%(11%)>0%TH U%(15
%)=U%(11%)
119 FOR I=8%TO15%
120 IF J%=1%TH U%(I)=U%(I
)*2%
121 IF I=11%ANDU%(I)>0%TH
 U%(I)=U%(I)+50%
122 ? AT((I-6%)*3%,22%)U%
(I)AT((I-6%)*3%+1%,23%);
123 IF V%(I,NK)=0%TH?"-";
EL?"+";
124 NEXT I
125 LINE (10%,239%)-(250%
,239%),2%
126 J%=J%+1
127 IF J%<4%TH59
128 IF NK<=QQ-NN TH190
129 B%=15%
```

```
130 A%=1%
131 CIRCLE (35%,184%),4%
132 B¤=""
133 B¤=INKEY¤
134 IF B¤=""TH133EL137
135 A¤="12345"
136 IF J%=1%TH65EL211
137 E=ASC(B¤)
138 B%=A%
139 IF E=25ANDA%<>6%ANDA%
<>15%TH B%=A%+1%
140 IF E=8ANDA%<>7%ANDA%<
>1%TH B%=A%-1%
141 IF E=26%TH IF A%>6%AN
DA%<12%THB%=A%-6%EL IF A%
>11%TH B%=6%
142 IF E=27%ANDA%<7%THB%=
A%+6%
143 IF E=10%ORE=32%TH152
144 IF E>32%TH132
145 IF A%>6%TH CI(11%+24%
*(A%-6%),234%),4%,0%EL CI
(11%+24%*A%,184%),4%,0%
146 IF B%>6%TH CI(11%+24%
*(B%-6%),234%),4%EL CI(11
%+24%*B%,184%),4%
147 A%=B%
148 GOTO 132
149 FOR S%=1%TO10000%
150 NEXT S%
151 RETURN
152 IF V%(A%,NK)=1%TH132
153 IF A%<7%TH X%(NK)=X%(
NK)+U%(A%)
154 Z%(NK)=Z%(NK)+U%(A%)
155 L%(NK)=U%(A%)
156 V%(A%,NK)=1%
157 R%=0%
158 FOR I=1%TO15%
159 IF V%(I,NK)=1%TH R%=R
%+1%
160 NEXT I
161 IF X%(NK)<0%ANDG%=15%
TH Z%(NK)=Z%(NK)-50%
162 ? AT(19%,11%)X%(NK);"
    "Z%(NK)
163 FOR I=1%TO5%
164 GOSUB 149
165 NEXT I
166 NEXT KK
167 FOR I=1%TOQQ
168 FOR J=I TOQQ
169 IF (L%(PX(I))<L%(PX(J
))ANDG%<15%)OR(G%=15%ANDZ
%(PX(I))<Z%(PX(J)))TH173
170 NK=PX(I)
171 PX(I)=PX(J)
172 PX(J)=NK
173 NEXT J
174 NEXT I
175 CLS
176 ? AT(10%,3%)"И Т О Г
И :"AT(15%,1%)G%AT(11%,6%
)"игра вс  шк пр общее"AT
(0%,8%)
177 FOR I=1%TOQQ
178 NK=FIX((2%*L%(I)-L%(P
X(1%))-L%(PX(QQ)))/5%)
179 IF NK<=0%TH NK=0%ELNK
=NK+1%
180 ? I;". ";J¤(I)TAB(11%
)L%(I)TAB(15%)Z%(I)TAB(19
%)X%(I)TAB(23%)NK TAB(28%
)Z%(I)+NK
181 Z%(I)=Z%(I)+NK
182 NEXT I
183 FOR I=1%TO15%
184 GOSUB 149
185 NEXT I
186 NEXT G%
187 ? "          Выйграл ";
J¤(PX(QQ))
188 GOSUB 149
189 GOTO 188
190 A%=0%
191 B%=0%
192 FOR I=1%TO15%
193 IF U%(I)=>0%ANDV%(I,N
K)=0%ANDRE(I)<=U%(I)ANDP%
(A%)>P%(I)TH A%=I
194 IF V%(I,NK)=0%TH B%=I
195 NEXT I
196 IF A%=0%ANDJ%>3%TH213
197 IF A%=0%OR(A%<7%ANDJ%
<4%)TH200
198 IF A%>6%TH CI(11%+24%
```

```
*(A%-6%),234%),4%EL CI(11
%+24%*A%,184%),4%
199 GOTO 153
200 FOR I=7%TO11%
201 IF U%(I)>0%TH A%=I-6%
202 NEXT I
203 IF A%=1%TH135
204 A%=U%(A%+6%)/A%
205 IF J%=2%TH A%=A%/2%
206 IF B%<7%ANDA%<>B%ANDV
%(A%,NK)>0%TH A%=B%
207 A¤=""
208 FOR I=1%TO5%
209 IF B%(I)<>A%TH A¤=""+
A¤+CHR¤(I+48%)
210 NEXT I
211 ? "?";A¤
212 GOTO 65
213 FOR I=1%TO15%
214 IF V%(0%(I),NK)=0%TH
A%=0%(I)
215 NEXT I
216 GOTO 198
```



А.Н.Лашков
(г.Капустин Яр)

Игра ПОКЕР (поправки к программе)

В седьмом номере за 1990 г. научно-популярной серии "Вычислительная техника и ее применение" был опубликован текст игровой программы "Покер". К сожалению, по многим причинам в программе допущен целый ряд опечаток.

Строки с 30-й по 200-ю рисуют на экране игровую таблицу. Трудность здесь может возникнуть из-за недостающих пробелов. Нижняя часть таблицы разбита на пять полей, которые разделяются символами ':'. Длина полей должна быть следующей:

Первое поле — длина 6 символов;
Второе — 7 символов;
Третье — 6 символов;
Четвертое — 11 символов;
Пятое — до конца строки.

Строки 30...50 рисуют рамку таблицы и вместо символа '@' был применен символ "Забой" (черный квадрат), который не удалось распечатать.

Вот полный (будем надеяться), перечень опечаток с пояснениями:

Срока 60 — оператор CUR имеет координаты 14,23.
Во второй функции CHR$ пропущен "$".
90 — перед символами S1 нужен пробел, между символами F и L нет пробела.
Второй оператор PRINT имеет вид:
PRINT STR$(I);":";.
Строка 250 — необходимо добавить NX = 1.
860 — оператор CUR имеет координаты 1,2.
870 — второй оператор PRINT печатает 5 пробелов.
910 — функция MID$ имеет вид:
MID$(A$,I,1).
1190 — вместо переменной F$ должна быть A$.
1460 — оператор PRINT имеет вид:
PRINT STR$(S(NI));" "
1580 — оператор PRINT печатает 11 пробелов.
1770 — после слова "игроку" строка имеет вид строки 1750.
2040 — условный оператор IF имеет вид:
IF A(I) = 5 ...
2100 — S5(NI) = A1 ...
2320 — второй оператор FOR имеет вид:
FOR J = 2 + Y TO 5.
2330 — операция отношения > =
2540 — вместо NETI, конечно же, NEXT.
2650 — исправьте 1560 на 1490.
2780 — исправьте 1 на I. Должно быть W = I
2820 — IF A(2) = A(I) ...
2990 — ... IF A(3) = A(I) ...
3570 — IF H(2) + H(3) ...
3990 — ... THEN Y = I
4270 — GOTO 4070

И, наконец, в листинге нет трех последних строк. Они имеют вид:

```
4380 IF S$(NI) = "1" THEN Z$ = "SU" : GOTO
     1350
4390 S$(NI) = "1" : Z(NI) = A(1)
+ A(2) + A(3) + A(4) + A(5) :
CUR 52,21-NI
4400 PRINT STR$(Z(NI)) : GOTO 1490
```

И в заключение несколько слов о

программе. Токенизированный текст программы "ПОКЕР" занимает в ОЗУ около 12 К. Если учесть, что сначала в память загружается интерпретатор БЕЙСИКа, то объем используемой памяти увеличивается до 20 К.

Программа сконструирована на компьютере "Радио-86" (С ОЗУ 32К) с использованием интерпретатора БЕЙСИКа "МИКРОН". Относительно легко программу можно адаптировать на компьютеры, допускающие отображение на экране дисплея 64-х символов в строке. Если при адаптации программы возникнут затруднения, то можно обратиться к журналам "Радио" № 1, 2, 3 за 1985 г. и № 8 за 1988 г., где подробно описаны все операторы и функции использованной версии БЕЙСИКа. При необходимости любой оператор или любую функцию можно заменить операторами, используемыми в вашей версии интерпретатора.

При набивании программы внимательно следует отнестись к использованию буквы О и цифры 0, которые отличаются только шириной (а в строке 3000 даже одинаковые). Особую опасность представит использование арабской цифры 1 вместо латинской буквы I, так как в этом случае программа может быть синтаксически верной, но работать будет неправильно из-за нарушения алгоритма.

****

Ленинградский производственный кооператив 'ОСТ' предлагает

-программное обеспечение для пользователей компьютеров БК-0010,0010.01, ДВК-2М,3М, УКНЦ, классов информатики КУВТ-86, КУВТ-УКНЦ;

Кооператив располагает обширным банком игровых, учебных, системных,прикладных программ.

Расценки значительно ниже государственных.

Списки программ, при указании типа ЭВМ, высылаются бесплатно.

Для пользователей БК-0010, 0010.01:

-программную разработку, позволяющую на каждой кассете пользователя размещать в 3 раза больше программ по сравнению с традиционным способом записи, при многократном увеличении надежности (магнитофон любого типа);

-недорогой телефонный модем на базе БК-0010, 0010.01.

Переписка: *189510, Ленинград, а/я 649.*

****

Научно-производственный кооператив "МОНИТОР" НАПОМИНАЕТ:

аппаратно-программные отладочные средства для однокристалльных ЭВМ — грандиозные возможности сегодня и неограниченные перспективы в будущем!!! У нас вы можете приобрести необходимый пакет программ для автоматизированного проектирования ПО практически для любой ОЭВМ!

Также мы поставляем:

- универсальный программатор, работающий пбд управлением ПЭВМ. Предназначен для программирования микросхем К573РФ2(РФ5), К573РФ3, К573РФ4(РФ6), К1816ВЕ48, К556РТ4, К556РТ5, К556РТ7, К556РТ16, К155РЕ3, INTEL 27128, 27256;
- системное и игровое ПО для ПК "ЛЬВОВ";
- техническую документацию на музыкальные звонки с 16 мелодиями (описание, схемы, копии фотошаблона, таблица прошивки ППЗУ и др.);
- практически любое системное и разнообразное прикладное ПО для ПЭВМ типа IBM PC.

Наш адрес: 290044, г.Львов-44, а/я 8863, НПК "МОНИТОР"

Справки по тел.: 35-35-79, 34-32-12 с 8 до 17 часов

34-29-42, 42-86-36 с 18 до 8 часов

Наши возможности и ваши желания - гарантии успеха !

С 41 **Системы коллективного пользования**. — М.: Знание, 1990. — 48 с. — (Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение"; № 12).

ISBN 5-07-001851-5

20 к.

В этом выпуске читатель познакомится с многотерминальными системами, системами виртуального доступа, сетями ЭВМ и получит рекомендации по выбору типа системы, соответствующей его производственным задачам.

Материал рассчитан на широкий круг читателей.

2404000000 **ББК 32.97**

Научно-популярное издание

СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

Гл.отраслевой редактор *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл.редактор *Н. А. Васильева*
Художник *В. Н. Конюхов*
Худож.редактор *И. А. Емельянова*
Техн.редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ № 11558

Подписано к печати 16.11.90. Формат бумаги 70x100 1/16. Бумага офсетная. Печать офсетная.Усл.печ.л.3,90. Усл.кр.-отт.8,45.Уч.-изд.л.4,18. Тираж 71574 экз. Заказ 1779. Цена 20 коп. Издательство "Знание". 101835, ГСП, Москва, Центр, проезд Серова,д.4. Индекс заказа 904712. Отпечатано с оригинал-макета издательства "Знание" на ордена Трудового Красного Знамени Тверском полиграфическом комбинате Государственного комитета СССР по печати. 170024, г.Тверь, пр.Ленина,5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

СОПК 5-27

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЕ ПРИМЕНЕНИЕ

Люди с психологией машинопоклонников часто надеются, что в высокоавтоматизированном мире потребуется меньше изобретательности, чем в наше время; они надеются, что мир автоматов возьмет на себя наиболее трудную часть нашей умственной деятельности — это явное заблуждение.

Н. Виннер

Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4